Navy Personnel Research and Development Center

San Diego, CA 92152-6800 TN-90-16 April 1990

2

The MS-DOS Device Handler Interface

Brian Van de Wetering
Brian Thomason

NPRDC-TN-90-16 April 1990

The MS-DOS Device Handler Interface

Brian Van de Wetering
Brian Thomason
Systems Engineering Associates
San Diego, California 92109

Reviewed and released by
Wallace H. Wulfeck II
Director, Training Technology Department

Approved for public release;
distribution is unlimited

Navy Personnel Research and Development Center
San Diego, California 92152-6800

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
April 1990

3. REPORT TYPE AND DATE COVERED
Interim--Sep 88-Feb 90

4. TITLE AND SUBTITLE
The MS-DOS Device Handler Interface

5. FUNDING NUMBERS
Program Element 0604722J
Work Unit 99-PJ1-90-006

6. AUTHOR(S)
Brian Van de Wetering and Brian Thomason

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Navy Personnel Research and Development Center
San Diego, California 92152-6800

8. PERFORMING ORGANIZATION REPORT NUMBER
NPRDC-TN-90-16

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of the Assistant Secretary of Defense
(Force Management and Personnel) (Room 3E808, Pentagon)
Washington, DC 20301-0000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Many vendors produce high-performance, low-cost training hardware, but bundle their products with proprietary software interfaces. Because these interfaces are proprietary, courseware and authoring systems written to operate on one set of hardware will not run on a competitor's hardware. Expensive reprogramming is needed to adapt to new hardware. These reprogramming costs can be eliminated by adopting standard software interfaces. The objectives of this effort were to describe and develop a standard software interface that will allow training systems to be assembled from separate "plug-and-play" components in the same way that stereo systems can be assembled from separate speakers, amplifiers, and other components. The Portable Courseware (PORTCO) architecture consists of two interfaces, the Device Services Interface and the Device Handler Interface. It also contains three layers: application, routing and configuration, and device handler. This architecture should allow applications software to run on any compliant set of hardware components. The series of reports describing the PORTCO architecture should direct development of portable MS-DOS applications and standard peripheral device handlers. This report describes the Device Handler Interface and should be of primary interest to device manufacturers and system vendors who must develop device handler software.

Keywords,

14. SUBJECT TERMS
Courseware portability, computer-based training, interactive courseware, virtual device interface (RH)

15. NUMBER OF PAGES
116

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

Foreword

This document describes the *device handler interface,*[1] a component of an architecture for interactive courseware delivery systems. It was developed under the sponsorship of the Office of Secretary of Defense (Force Management and Personnel). Mr. Gary Boycan was the technical sponsor. Funding was provided under Program Element 0604722J, Work Unit 99-PJ1-90-006.

This document was developed under the technical supervision of Dr. Raye Newmen, Dr. Wallace H. Wulfeck, and Mr. Walter F. Thode of the Navy Personnel Research and Development Center (NPRDC). This report itself was written by Systems Engineering Associates under Contract N66001-88-D-0054, Delivery Order 7J30.

This architecture was developed as part of an effort to complete a reference implementation of a courseware portability specification. The implementation is scheduled for later this year. Comments on the architecture described here are solicited from all interested parties. The specification will then be submitted for consideration for official adoption by the Department of Defense and the National Institute of Standards and Technology.

Point of contact at NPRDC is Walter F. Thode (619) 553-7703 or AUTOVON 553-7703.

WALLACE H. WULFECK II
Director, Training Technology Department

[1] Special terms appear in *italics* when first used, and are defined in Section 8.

Summary

Background

Over the next several years the Federal Government will invest millions of dollars to develop training materials for delivery on computer-based interactive training systems. To support this investment, Federal agencies will acquire a variety of computers and peripheral devices. This hardware will host several operating systems and authoring system software to speed courseware development.

Many vendors produce high-performance, low-cost training hardware, but bundle their products with proprietary software interfaces. Because these interfaces are proprietary, courseware and authoring systems written to operate on one set of hardware will not run on a competitor's hardware. Expensive reprogramming is needed to adapt to new hardware. These reprogramming costs can be eliminated by adopting standard software interfaces.

Objectives

The objectives of this effort were to describe and develop a standard software interface that will allow training systems to be assembled from separate "plug-and-play" components in the same way that stereo systems can be assembled from separate speakers, amplifiers, and other components.

Approach

The Portable Courseware (PORTCO) architecture consists of two interfaces, the Device Services Interface and the Device Handler Interface. It also contains three layers: application, routing and configuration, and the device handler. This architecture should allow applications software to run on any compliant set of hardware components.

Results and Conclusions

This report is the third in a series of five reports; it describes the Device Handler Interface and should be of primary interest to device manufacturers and system vendors who must develop device handler software. The first report provides an overview of the PORTCO architecture and should be of interest to all who are concerned with computer-based training. The second report describes the Ms-DOS Device Services Interface and is intended primarily for programmers who want to develop portable application software. The fourth report is intended for system vendors and describes the design of the

first PORTCO routing and configuration program. The fifth report is intended as additional support for those who must develop complaint MS-DOS device handlers.

Recommendations

1. The reports describing the PORTCO architecture should direct development of portable MS-DOS applications and standard peripheral device handlers.

2. The architecture should serve as a foundation for compliance with the Interactive Video Industry Association's "Recommended Practices for Interactive Video Portability." The architecture should motivate development of specific applications and device handlers that adhere to this specification.

3. Feedback about problems and suggested improvements should be forwarded to the Navy Personnel Research and Development Center, Code 152.

Table of Contents

List of Tables

List of Figures

1. Purpose and Scope

This document describes the DHI, the boundary between the *Routing and Configuration* (R&C) layer and the *device handler layer* of the PORTCO architecture. It is intended to guide development of PORTCO device handlers by peripheral manufacturers, and to provide background information for developers of R&C-layer software. This is one of several publications, listed in Table 1, describing the PORTCO architecture in an MS-DOS[2] environment.

Table 1
This specification is one of a collection of PORTCO documents.

Title	Description
A Portable Courseware Architecture	A top-level description of the PORTCO architecture.
The MS-DOS Device Services Interface	A specification for the MS-DOS interface between the PORTCO architecture's Application layer and its R&C layer. This specification guides the development of compliant applications and compliant R&C programs.
The MS-DOS Device Handler Interface	A specification for the MS-DOS interface between the PORTCO architecture's R&C layer and its Device Handler layer. This specification guides development of compliant MS-DOS device handlers and compliant MS-DOS R&C programs.
MS-DOS Routing and Configuration Program Design	A specification for the first implementation of compliant MS-DOS R&C-layer software, expected to serve as an example for future implementations.
Guidelines for Implementing MS-DOS Device Handlers	A collection of informal guidelines and examples to support the development of compliant MS-DOS device handlers.

All readers should be familiar with *A Portable Courseware Architecture* (Thomason, Van de Wetering, & Booth, 1990), the first publication listed in Table 1, and should be accomplished MS-DOS system programmers with a sound understanding of "C", MS-DOS system services, and assembly language.

All device handler developers should carefully study Sections 2 through 6 in this publication before designing their software. Readers should also study the portion of Section 7 and the appendix that describes their handler's logical device (e.g., Section 7.1 and Appendix A describe logical

[2] MS-DOS is a registered trademark of Microsoft Corporation.

videodisc players).

Developers of device handlers will also benefit by reviewing Thomason and Van de Wetering (1990) for a description of applications use of device handlers. Van de Wetering and Thomason (1990) provides detailed examples of device handler construction, and should also be of interest. Table 2 highlights the location of answers to common questions.

Table 2
Reader's questions are answered in various sections.

Section References	Device Handler Developers' Questions
1	What parts of this publication do I need to read? How about other documents?
7, Appendices A-D	What functions must my device handler provide?
3, 5, Appendices A-D	How does the R&C program call these functions?
3	How must my device handler be packaged for delivery?
4	What memory areas can my device handler use?
3, 4	How is my device handler loaded into memory and initialized?
4, Appendices A-D	How should I report errors? Can I write to the system console?
4	How can I display a copyright notice?
4	Can command-line options be passed to my handler?
4, 5.1, 5.3, Appendices A-D	How do I communicate alerts to the R&C program?
2, 5.1, 5.3	How does a device handler alert an application when a hardware interrupt occurs?
4	When should a device handler install hardware ISRs?
3, 5.3	Who "owns" an alert packet's memory?
5.1, 5.3	How do device handlers pass alerts to the R&C program?

2. Background

The DHI is the boundary between the PORTCO architecture's R&C layer and its device handler layer. (Figure 1 illustrates the PORTCO architecture and the DHI's position within it.) The R&C layer is implemented as an MS-DOS TSR program, named the R&C program, and the device handler layer is implemented as MS-DOS executable files called device handlers. So the DHI defines the boundary between PORTCO device handlers and the R&C program.

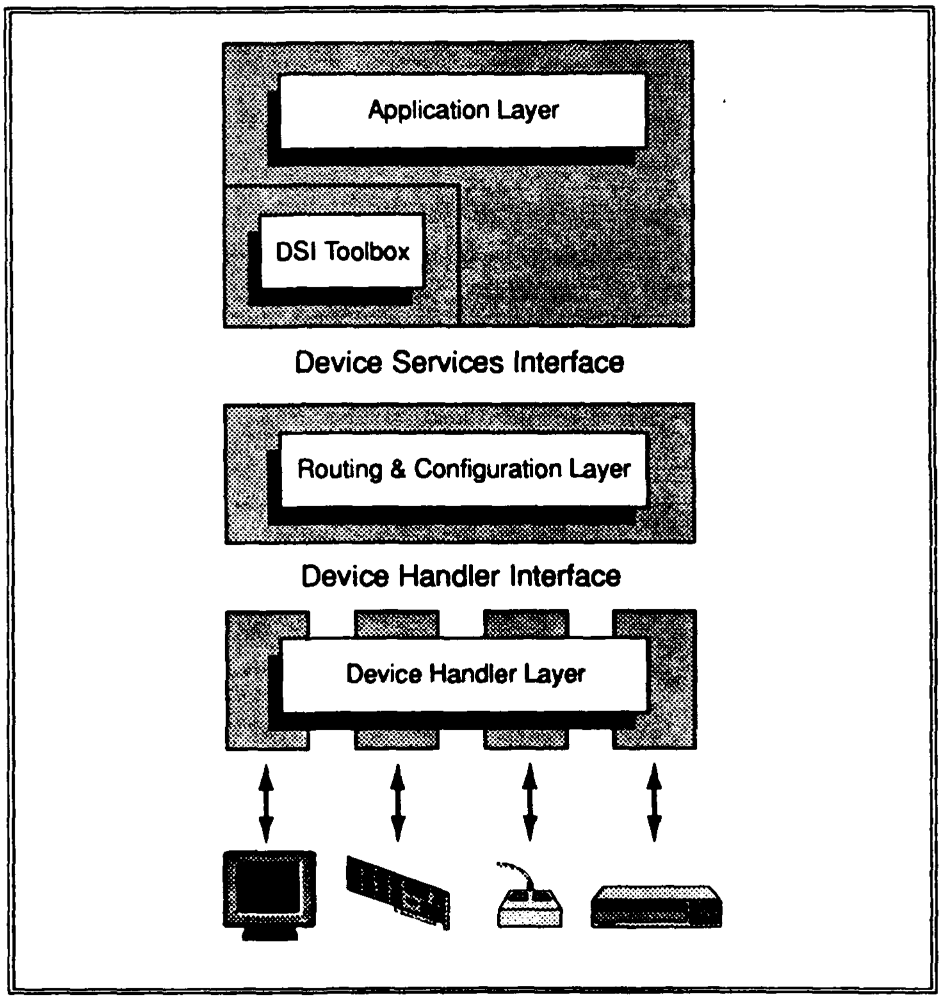


Figure 1
The PORTCO architecture contains three layers and two interfaces.

Each device handler implements the services of a single logical device on a particular kind of peripheral. For example, a device handler might implement logical videodisc services on an Hitachi videodisc player, or logical locator services on a Microsoft mouse.

Device handlers are loaded into memory and initialized by the R&C program using the conventions detailed in Section 3. Thereafter, the R&C program exchanges packets with device handlers by placing the packet's address in a register and issuing an assembly language call. (Packet routing conventions are described in Section 5.1.) Device handlers respond to service requests and issue alerts when critical hardware interrupts occur. When system operation concludes, the R&C program invokes each handler's termination service (specified in Appendices A through D), then returns memory used by the handler to MS-DOS.

All PORTCO device handlers provide the five common services listed in Table 3. Each handler also provides the core services in its device class. Support for extended services is optional; developers may choose to implement any of the extended services in this publication. All services are listed in Section 7 and specified in Appendices A through D.

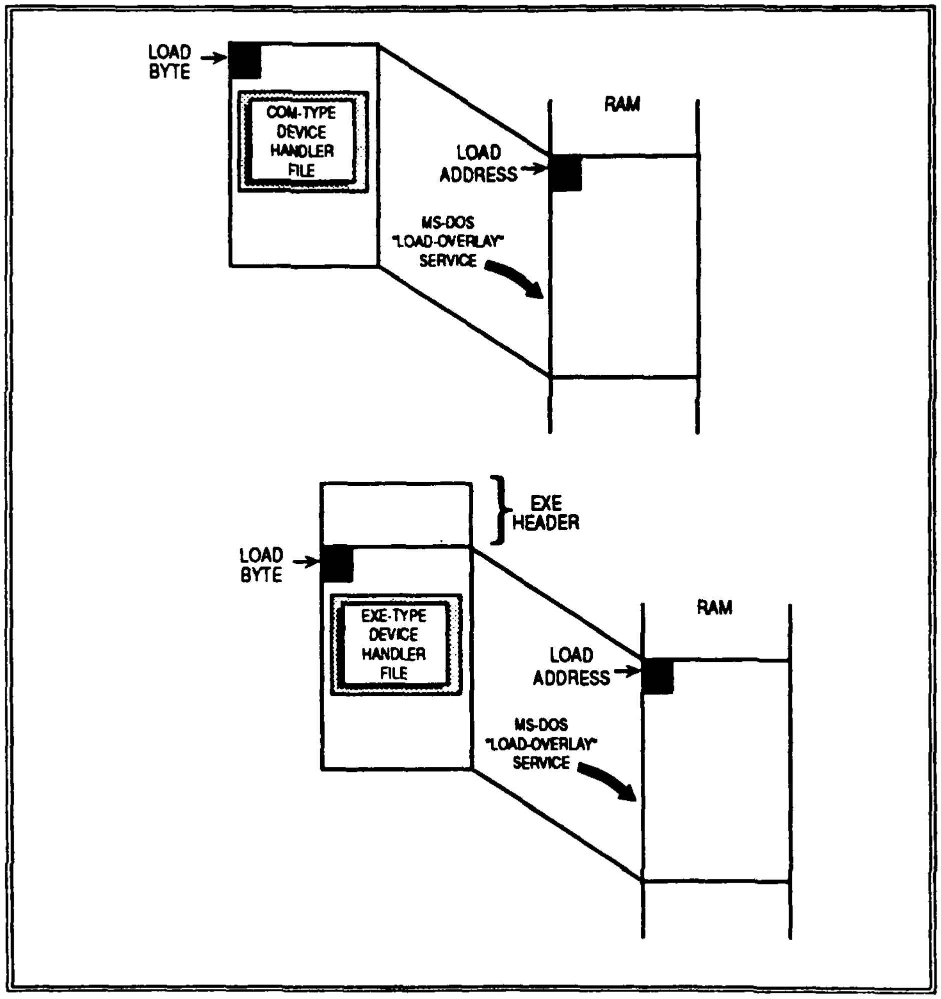
Table 3
All device handlers share five common services.

Name	Identification
Initialize Device Handler	0
Terminate Device Handler	1
Services Query	2
Set Alert Mask	3
Disable All Alerts	4

3. Loading Device Handlers

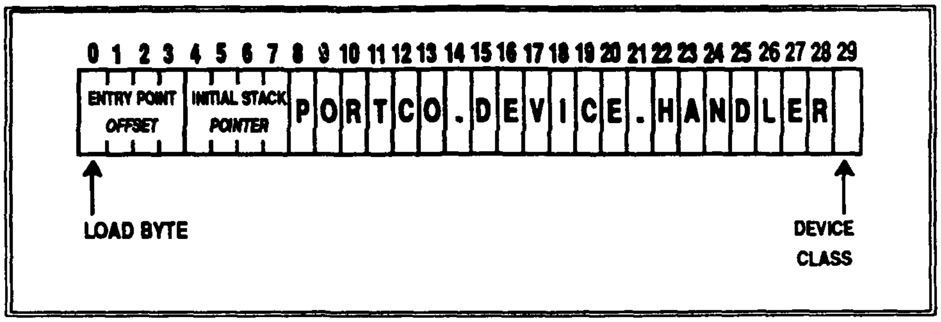
PORTCO device handlers are implemented as MS-DOS COM- or EXE-type files with respective filename extensions of ".DVC" and ".DVX". MS-DOS provides a system function called "load overlay" (int 21h, function 4Bh, AL=3) that transfers such files into RAM at a user-defined *load address* for subsequent execution. The byte in each file that is deposited at the load address by MS-DOS's load overlay function is called the file's *load byte*. In COM-type files, this is the file's first byte. In EXE-type files, this byte immediately follows the file's header (Duncan (1986) describes EXE-type file headers). Figure 2 illustrates the relationship between a device handler's load byte and its load address.

Figure 2
A device handler's load byte is deposited at its load address by MS-DOS's "load overlay" service.


LOAD BYTE
COM-TYPE DEVICE HANDLER FILE
RAM
LOAD ADDRESS
MS-DOS "LOAD-OVERLAY" SERVICE
EXE HEADER
LOAD BYTE
EXE-TYPE DEVICE HANDLER FILE
RAM
LOAD ADDRESS
MS-DOS "LOAD-OVERLAY" SERVICE

A device handler's load byte and its following 29 bytes identify it as a PORTCO handler, specify its device class, identify its entry point, and specify the values of SS:SP for the first call to the handler. Figure 3 illustrates the format and content of these bytes.

Figure 3
30 bytes identify a device handler.



As Figure 3 suggests, a device handler's load byte and its three following bytes contain an unsigned, 32-bit integer: the handler's entry point offset. The handler's entry point is calculated by adding this integer to the handler's load address. The R&C program passes control to tnis ac'dress when it sends a packet to the handler.

The handler's next four bytes contain the initial value of the handler's stack pointer (SS:SP). The R&C program loads these values into the SS and SP registers before calling the handler the first time. The R&C program ensures that the handler's stack is properly set up each time it calls the handler. Because COM-type files may not contain segment references, the R&C program ignores the stored value of SS in COM-type files and loads SS with the segment value of the device handler's entry point.

The handler's next 21 bytes contain the uppercase ASCII string, "PORTCO_DEVICE_HANDLER," and identify the file as such. The R&C program will not load device handlers without this identification string.

The 29th byte following a device handler's load byte contains an unsigned 8-bit integer identifying the handler's device class. Table 4 specifies this byte's value for each logical device.

Table 4
An unsigned, 8-bit integer identifies each device class.

Device Class	Value
Videodisc Player	0
Video/Graphics Overlay	1
Locator	2
Audio Management	3

COM-type device handlers are loaded into memory by simply copying their disk file's contents. EXE-type handlers are loaded by stripping their MS-DOS header, loading the remaining portion into memory, and performing any relocation indicated by the file's relocation map. The MS-DOS "load overlay" system service (int 21h, function 4Bh, AL=3) may be used by the R&C program to load both file types.

4. Initialization

After loading a handler into memory, the R&C program invokes DHI service 0, "Initialize Device Handler." This service asks the device handler to initialize itself and its device. It must be the first request made of every device handler. If this service is invoked more than once, the device handler takes no action and returns a status code indicating that it has already completed this request.

The R&C program may pass an ASCII string to the device handler as part of this service. This string communicates initialization information to the handler, and is retrieved from the DSI configuration file (as described in Thomason and Van de Wetering (1990)). For example, the R&C program may pass the following initialization string to a videodisc player handler to tell it which communication port to use:

```
/PORT=COM1:
```

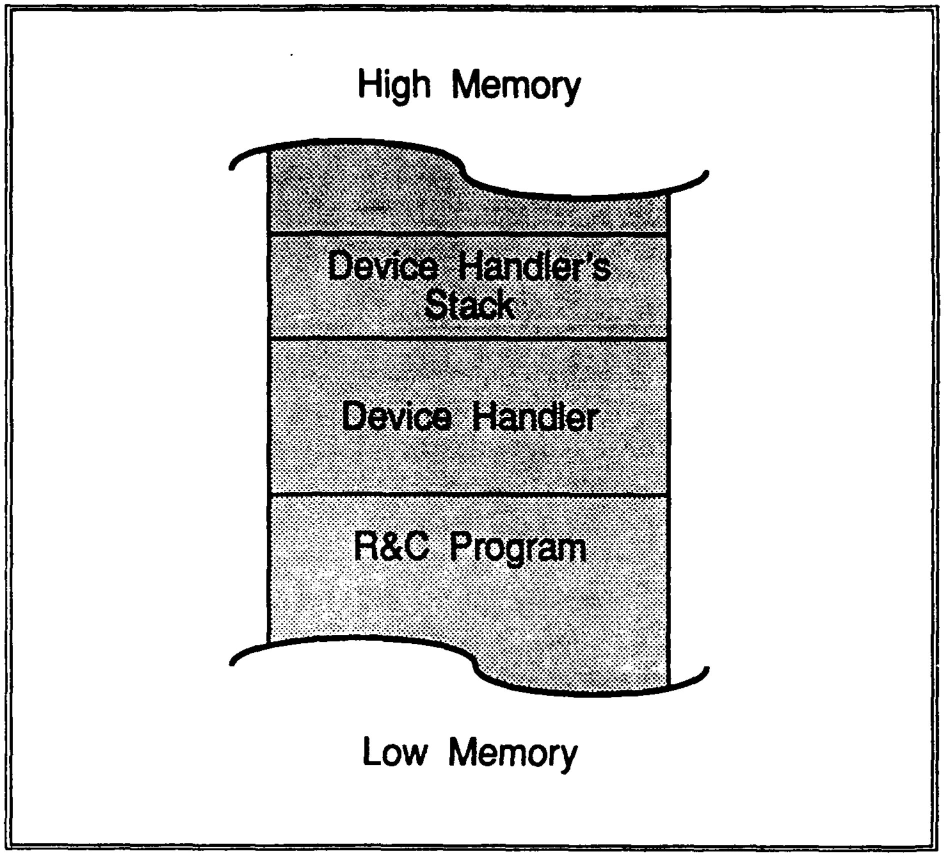
The syntax and semantics of initialization strings are dictated by device handler authors. Each device handler will respond to its own (possibly unique) collection of initialization strings.

In response to service 0, "Initialize Device Handler," the handler must tell the R&C program how much memory it needs for its stack (for generating alert packets or for any other purpose). The number of memory bytes available to the device handler for its stack is passed to it by the R&C program as a parameter of service 0. This memory is contiguous with and directly above the device handler. The device handler indicates its stack requirements by returning the next available segment address after the space it needs. Figure 4 illustrates locations of a device handler and its stack.

If the device handler needs to issue alerts, it must store the entry point address of the R&C program routine that routes alert packets. This address is passed by the R&C program as another parameter of service 0, "Initialize Device Handler." The device handler must also store its device class and number (from the packet's header), so that it can properly identify itself when issuing alerts. It must be sure to reserve enough memory for its alert packets, and it must install ISRs to catch any hardware interrupts it requires (ISRs should be installed without disrupting existing ISRs). Each device handler must also disable all alerts before completing its initialization service.

A device handler may present a copyright notice on the system console (standard output) during the first invocation of service 0. Such a notice should not appear at any other time.

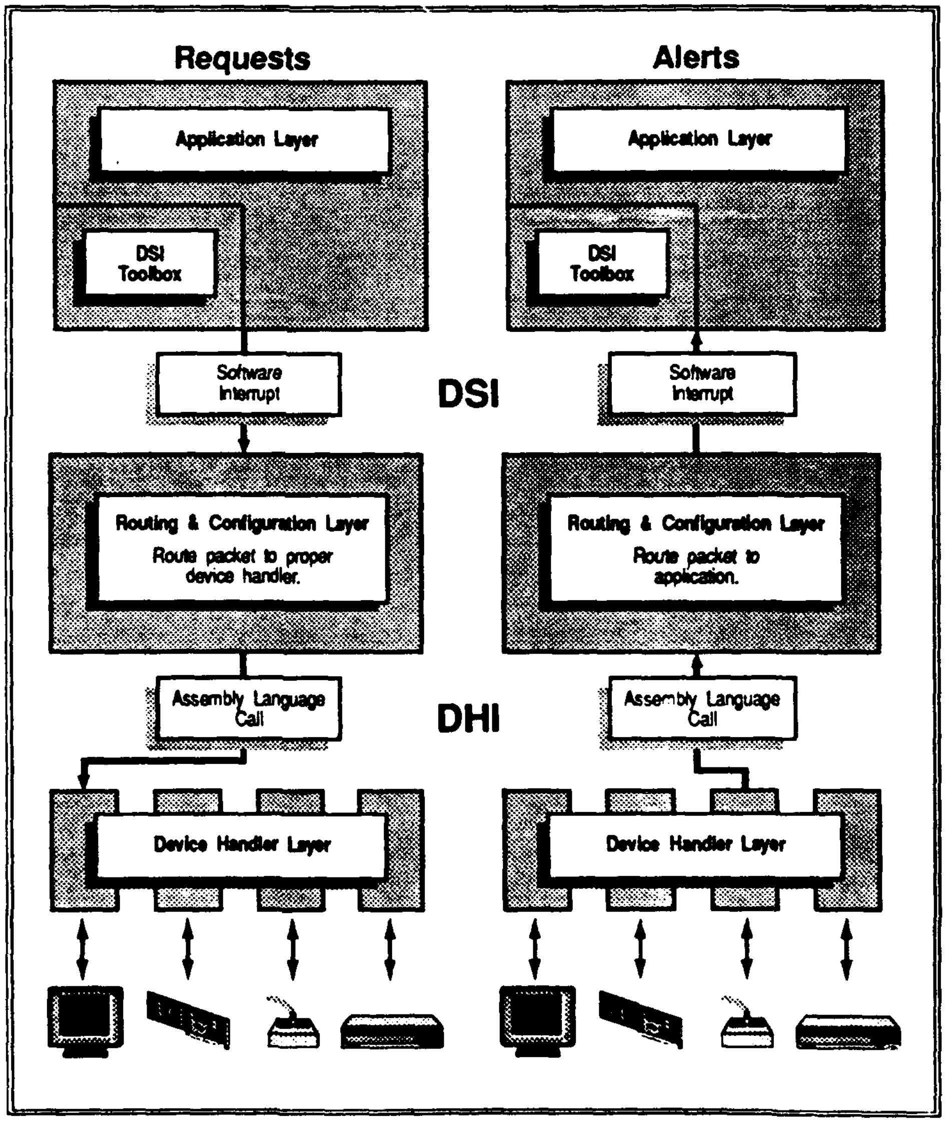
Figure 4
A device handler's stack is contiguous with the handler and directly above it.



5. Packets

The R&C program communicates with device handlers using contiguous blocks of data called packets. The R&C program passes packets to handlers to request service, and handlers pass packets to the R&C program to alert it of asynchronous device activity (i.e., critical hardware interrupts). Request packets solicit service, and alert packets announce device activity. Applications originate most (but not all) requests, and send them to device handlers via the R&C program. Similarly, alerts are sent to applications by the R&C program.. Figure 5 illustrates this packet-passing convention.

Figure 5
Packets are used to solicit services and issue alerts.



5.1 Packet Routing

To pass a request packet to a device handler, the R&C program places the packet's 32-bit address[3] into the ES:DX registers, restores the handler's stack pointer (SS:SP), then issues a far call to the handler's entry point. (The location of each handler's entry point is specified in Section 3.) Upon entry, the device handler saves any registers that it uses and restores them before issuing a far return to the R&C program. Upon return, the R&C program saves the handler's stack pointer (to prepare for the next request).

To pass an alert packet to the R&C program, a device handler places the packet's 32-bit address into the ES:DX registers and issues a far call to the R&C entry point identified during the handler's initialization (discussed in Section 4). Upon entry, the R&C program saves any registers it uses and restores them before issuing a far return to the handler. Because alerts are generated by hardware interrupts, device handler ISRs should minimize stack use, since they use whatever stack is active when the interrupt occurs.

5.2 Request Packets

Figure 6 illustrates a request packet. As this figure suggests, each packet consists of a header followed by a series of request blocks, and each packet may contain a sequence of requests.

The device class is the first packet field; it occupies one byte and is set by the application, using Table 5. (Data in all fields are formatted as unsigned integers unless otherwise specified.) The device number is the second field; it also occupies one byte and is set by the application. These two fields identify a logical device, the packet's destination (e.g., videodisc player 2).

The return status is the packet's third field; it occupies one byte and is set by the device handler to indicate successful completion of the packet's services. Zero here indicates success and a non-zero value indicates failure. When this field is non-zero, its value identifies the first request that failed. For example, if the packet's fifth request failed, this field would be 5. (Requests in the same packet following the failed request are not processed.)

The header's final field is the request count; it occupies one byte, is set by the application, and indicates the number of requests contained in the rest of the packet.

[3] All 32-bit addresses in this document are formatted using a standard 80x86 16-bit *segment* followed by a 16-bit offset.

Figure 6
A request packet contains a header and one or more request blocks.

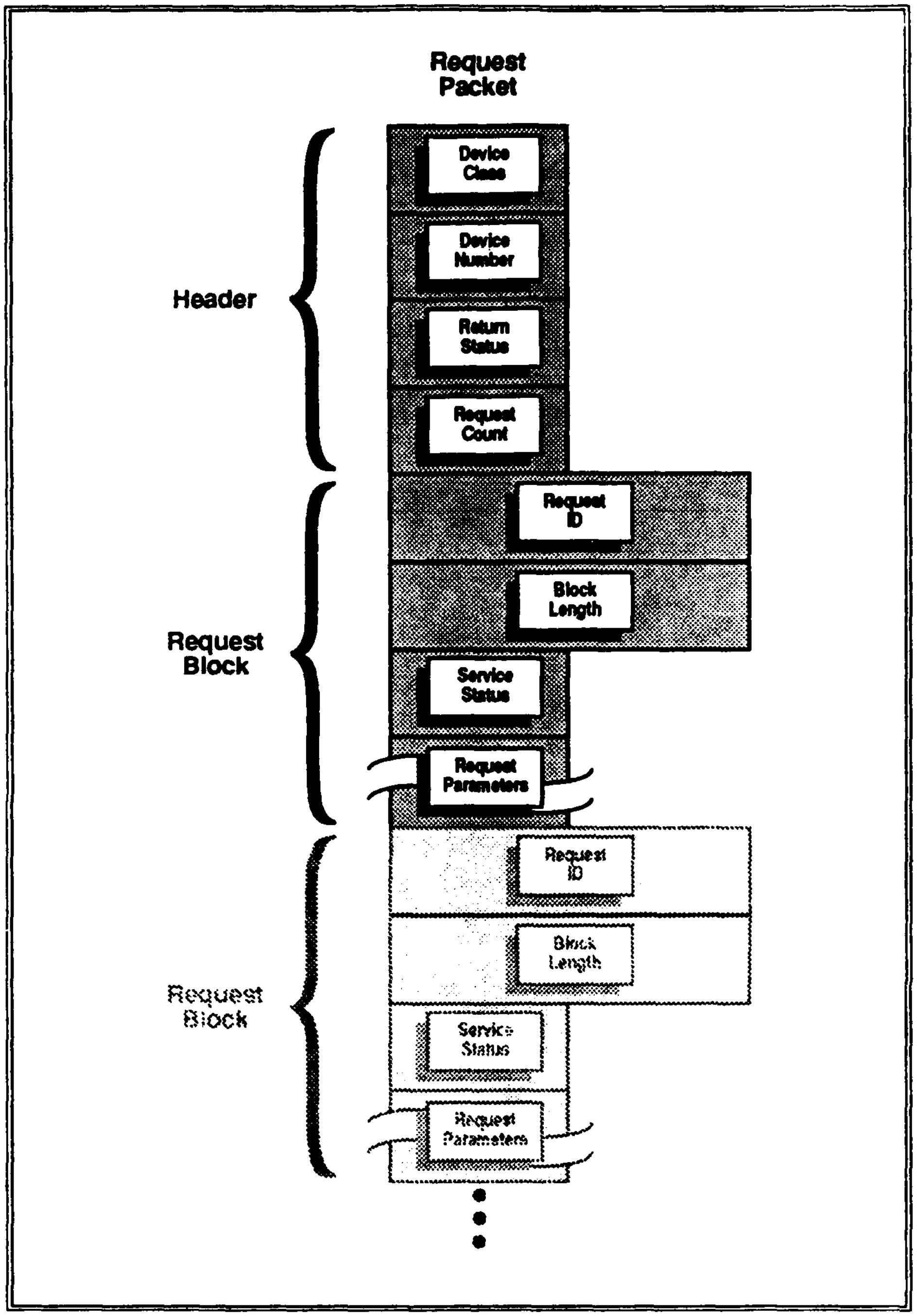


Table 5
Each device class is assigned a unique integer.

Device Class	Value
Videodisc Player	0
Video/Graphics Overlay	1
Locator	2
Audio Management	3

Each request block begins with a field to identify its service. This field occupies two bytes and is set by the application. Services for each logical device are labeled with a unique integer, which is used in this field. As examples, Table 6 lists a few videodisc player services and their identification numbers. Section 7 lists all DHI services.

Table 6
DHI services have integer labels.

Name	Service ID
Initialize Device Handler	0
Terminate Device Handler	1
Services Query	2
Set Alert Mask	3
Disable All Alerts	4
Player Status	5
Videodisc Status	6
Position Request	7
Spin Up/Down	8
Eject	9
Frame Search	10
Jump	11
Still	12
Chapter Search	13
Video On/Off	14
Audio1 On/Off	15
Audio2 On/Off	16

The next request-block field occupies two bytes and is set by the application to indicate the number of bytes in the block. This byte count should include the block's two-byte service identification.

The next field occupies one byte and is set by the device handler to indicate the service's success or failure. A zero value here indicates success, and a non-zero error code indicates failure. Each logical device uses a unique set of error codes. As examples, Table 7 lists some codes used by the logical videodisc player. Other error codes are listed in Section 7.

Service parameters make up the request block's remaining fields. The number and content of these fields vary with each request. For example, Figure 7 illustrates a request block for videodisc player's service 10, "Frame Search." All parameters in this case are set by the application. Figure 8 shows the request block for service 5, "Player Status." Parameters in this case are set by the device handler, and the block's status field indicates the handler's success at providing the requested information. (If a request for videodisc player status failed because the videodisc player was disconnected and did not respond, the status field would report this error, indicating to the application that the information returned was invalid.) Appendices A through D detail the service parameters of all DHI services.

Table 7
Each PORTCO device uses integer error codes.

Description	Error Code
Service successful	0
Unknown error	1
Device not available	2
Device handler busy	3
Player not responding	4
Player busy	5
Videodisc parked	6
Ejected	7
No videodisc in player	8
Invalid frame number	9
Invalid chapter number	10
Invalid jump offset	11
Chapters not available	12
Destination unreachable	13
Invalid speed selection	14
Player already at first frame	15
Player already at last frame	16
Remote control not present	17
Interrupted before completion	18
Already initialized	19
Must open with front panel	20
Player spun down	21

Figure 7
Request block parameters vary with each service, as indicated by this request block for "Frame Search."

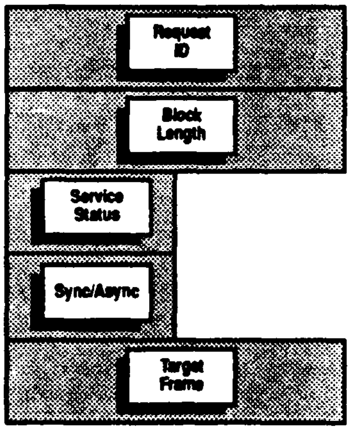
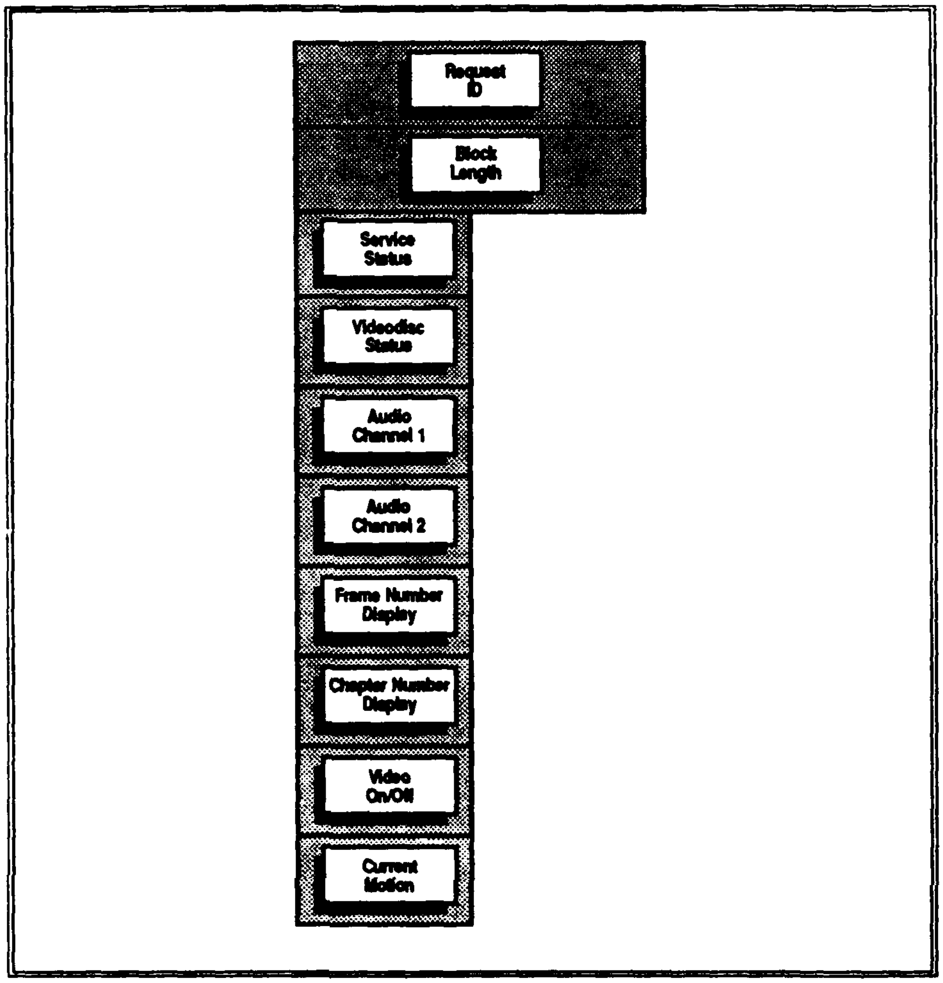


Figure 8
Request block parameters vary with each service, as indicated by this request block for "Player Status."



5.3 Alert Packets

Device handlers issue alerts in response to critical hardware interrupts. As illustrated in Figure 9, an alert packet consists of a header followed by a series of alert blocks. Each alert packet must reside in its originating device handler's memory region. Like a request packet, each alert packet may contain multiple alerts. All fields in this packet are set by device handlers and read by applications, and all fields are formatted as unsigned integers unless otherwise specified.

The device class is the packet's first field and occupies one byte. The device number is its second field and also occupies one byte. Together, these fields identify the logical device issuing the alert. The packet's third field, the alert count, occupies one byte. This field indicates the number of blocks in the rest of the packet.

An alert block's first field occupies two bytes and identifies a device activity. For example, Table 8 lists activities reported by a logical locator device. Section 7 and Appendices A through D describe alerts issued by all PORTCO logical devices.

Figure 9
Each alert packet communicates one or more alerts.

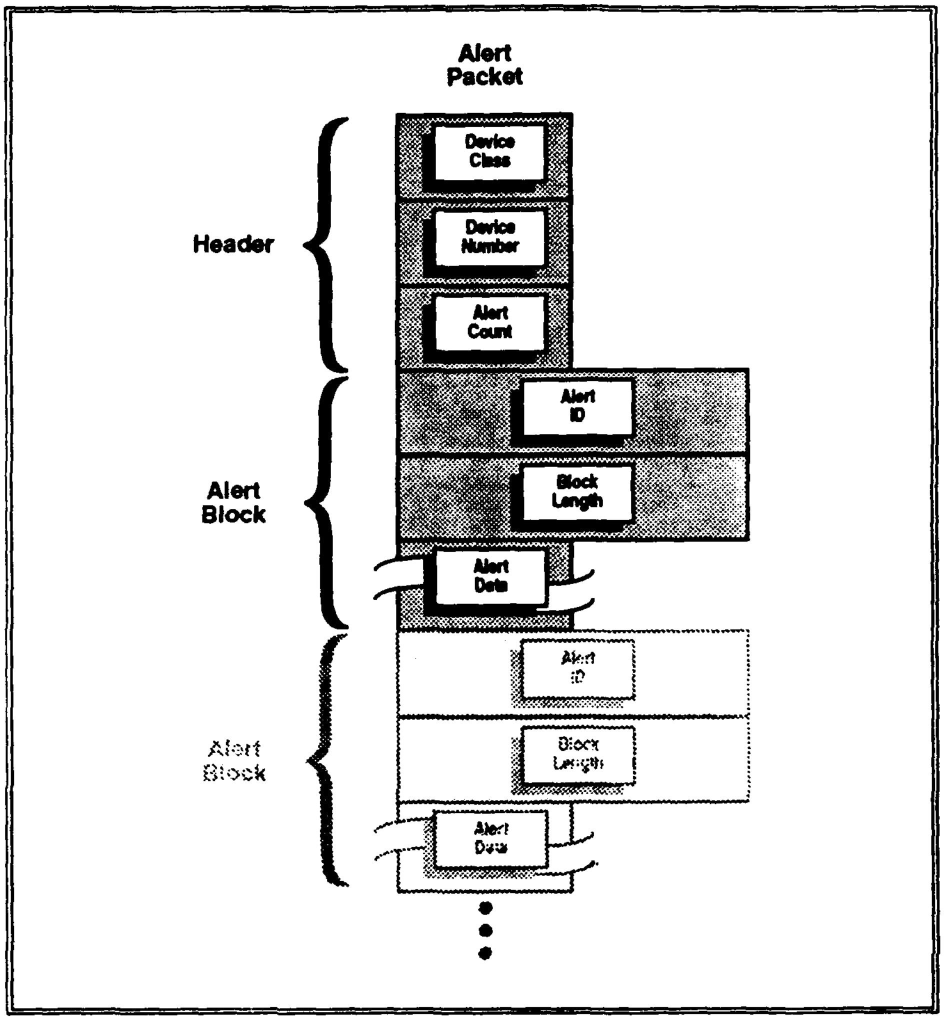
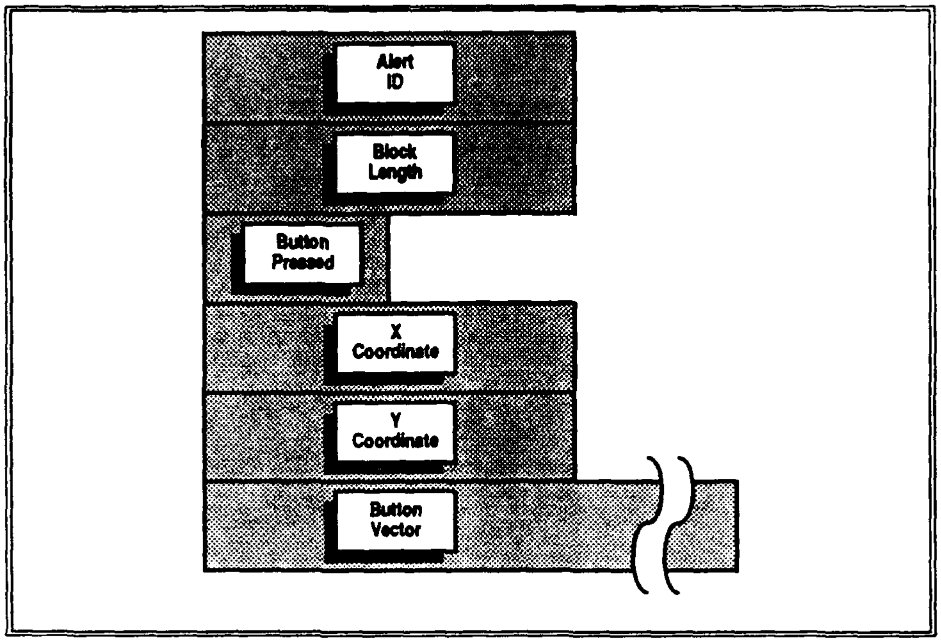


Table 8
A logical locator provides three alerts.

Name	Identification
Movement	0
Button Pressed	1
Button Released	2

An alert block's second field describes the number of bytes in the block (including the first two alert-ID bytes). Alert data make up the remaining fields in an alert block. The number and size of these fields depend on the alert being issued. For example, Figure 10 illustrates this block for the "Locator Button Pressed" alert. Appendices A through D describe these fields for all DHI alerts.

Figure 10
Alert block data vary with each alert.



6. Implementation Constraints

Device handlers are implemented as MS-DOS executable files, modified as described in Section 3 so they can be loaded and called by the R&C program. All device handlers must be relocatable. They may fit entirely in a single 64-k segment and contain only offset references relative to that segment, as is the case with .COM files, or they must contain appropriate relocation information so MS-DOS can relocate them when they are loaded, as is the case with .EXE files.

It must be possible for a device handler to be invoked (entered) by one program flow while an invocation of the same handler by another program flow is still executing. In such cases, neither invocation should fail or be corrupted. Such corruption or failure may occur if a device handler is interrupted while processing a request and the interrupt service routine itself requests service. For example, a videodisc handler might be interrupted by a locator alert ("Button Pressed"). In response, the application might generate another videodisc request to turn an audio channel on ("Audio 1 On/Off").

Device handler developers may use two approaches to satisfy this constraint. A device handler may be made truly reentrant by ensuring that each procedure invocation uses its own data area and that subsequent invocations do not destroy the context of previous invocations. Or the device handler may simply report a status code indicating that it is currently busy servicing a pending request and cannot process the current request.

7. Services and Logical Dɘvices

The DHI provides services for each logical device listed in Table 4 (page 6). As suggested in *A Portable Courseware Architecture* (Thomason & Van de Wetering, 1990), logical devices are fashioned from characteristics shared by several physical peripherals. The following sections present the conceptual models underlying each logical device and list all services each offers.

Some services may be executed synchronously or asynchronously. When a service is executed synchronously, control returns to the application when the service finishes. When a service is executed asynchronously, control returns as soon as the service is initiated. An alert is issued later when the service is finished.

7.1 Videodisc Player

A logical videodisc player presents motion video, still video, and audio output from a prerecorded source, under the control of computer software. A video frame is a single, still video picture that is labeled with an integer between 1 and 54,000 called a frame number. A logical videodisc player produces video and audio output in the modes presented in Table 9.

A logical videodisc player randomly accesses up to 54,000 frames of video information and provides two audio channels. Depending on the prerecorded source, the audio channels may make up a single stereo channel or may contain distinct audio tracks. The audio channels and the video output may be synchronized to produce television-like output. In addition to being controlled by computer software, a videodisc player may have a remote control unit or front-panel controls, allowing a user to control the device directly without a computer. Most videodisc players read prerecorded video and audio information from LaserVision format optical disks. These disks can be recorded in both Constant Linear Velocity (CLV) and Constant Angular Velocity (CAV) formats. CLV format disks typically do not allow random access to individual frames of video and so will not support most of the services in this device class.

Table 10 lists all videodisc player services and the page in this document where they are specified; Table 11 lists all alerts; and Table 12 lists all of the error codes returned by videodisc player services.

Table 9
A logical videodisc player has nine operating modes.

Mode	Description
Still	Presents a single still video picture. No audio is produced in this mode.
Normal forward	Presents motion video at approximately 30 frames per second, displaying frames in increasing numerical order. Audio is produced in this mode.
Slow forward	Presents motion video at between 1/4 and 3/4 of normal-speed motion, displaying frames in increasing numerical order. No audio is produced in this mode.
Fast forward	Presents motion video at between 2 and 4 times normal-speed motion, displaying frames in increasing numerical order. No audio is produced in this mode.
Scan forward	Presents motion video as fast as possible, displaying frames in increasing numerical order. No audio is produced in this mode.
Normal reverse	Presents motion video at approximately 30 frames per second, displaying frames in decreasing numerical order. No audio is produced in this mode.
Slow reverse	Presents motion video at between 1/4 and 3/4 of normal-speed motion, displaying frames in decreasing numerical order. No audio is produced in this mode.
Fast reverse	Presents motion video at between 2 and 4 times normal-speed motion, displaying frames in decreasing numerical order. No audio is produced in this mode.
Scan reverse	Presents motion video as fast as possible, displaying frames in decreasing numerical order. No audio is produced in this mode.

Table 10
A logical videodisc player provides 25 core services.

Name	Service ID	Core/Extended	Page [a]
Initialize Device Handler	0	Core	A-1
Terminate Device Handler	1	Core	A-2
Services Query	2	Core	A-3
Set Alert Mask	3	Core	A-4
Disable All Alerts	4	Core	A-4
Player Status	5	Core	A-5
Videodisc Status	6	Core	A-6
Position Request	7	Core	A-7
Spin Up/Down	8	Core	A-8
Eject	9	Core	A-9
Frame Search	10	Core	A-10
Jump	11	Core	A-11
Still	12	Core	A-12
Chapter Search	13	Core	A-13
Video On/Off	14	Core	A-14
Audio1 On/Off	15	Core	A-15
Audio2 On/Off	16	Core	A-16
Set Stop	17	Core	A-17
Set Inform Flag	18	Core	A-18
Remote Control On/Off	19	Core	A-19
Frame Display On/Off	20	Core	A-20
Chapter Display On/Off	21	Core	A-21
Chapter Play	22	Core	A-22
Motion	23	Core	A-23
Service Status Query	24	Core	A-24

[a] Pages appear in Appendix A of this publication.

Table 11
A logical videodisc player provides two alerts.

Name	Alert ID	Core/Extended	Page [a]
Service Complete	0	Core	A-25
Frame Arrival	1	Core	A-25

[a] Pages appear in Appendix A of this publication.

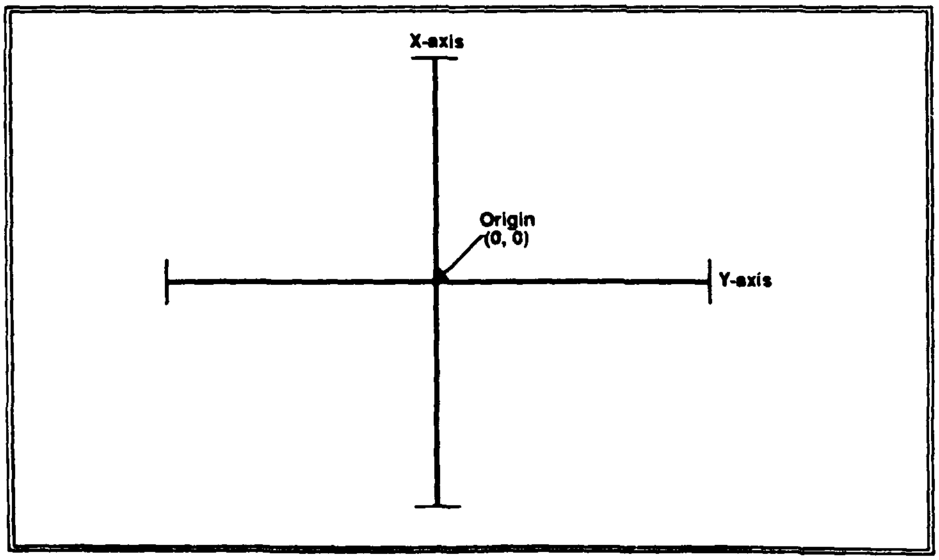
Table 12
A logical videodisc player provides 22 error codes.

Description	Error Code
Service successful	0
Unknown error	1
Device not available	2
Device handler busy	3
Player not responding	4
Player busy	5
Videodisc parked	6
Ejected	7
No videodisc in player	8
Invalid frame number	9
Invalid chapter number	10
Invalid jump offset	11
Chapters not available	12
Destination unreachable	13
Invalid speed selection	14
Player already at first frame	15
Player already at last frame	16
Remote control not present	17
Interrupted before completion	18
Already initialized	19
Must open with front panel	20
Player spun down	21

7.2 Locator

A locator is a logical device that reports two-dimensional position information. A locator reports its position as a pair of integers, called an X-coordinate and a Y-coordinate. These coordinates describe the locator's distance from two stationary, intersecting, perpendicular lines, called the X and Y axes. The point where the axes intersect is called the origin and has the coordinates (0, 0). Coordinate values on each axis can range from 32767 to -32768. Figure 11 illustrates a logical locator's coordinate system.

Figure 11
A logical locator uses a Cartesian coordinate system.



A locator may also report the state of from 0 to 256 logical buttons. Each button is labeled with an integer between 0 and 255, and a locator reports each button state as a binary value, pressed (1) or released (0). Each button state is represented as a bit within a bit-vector[4] (see Figure 12).

This bit vector is called the locator's button vector. The N'th bit in a locator's button vector represents the state of its N'th button. Figure 13 illustrates a locator's button vector.

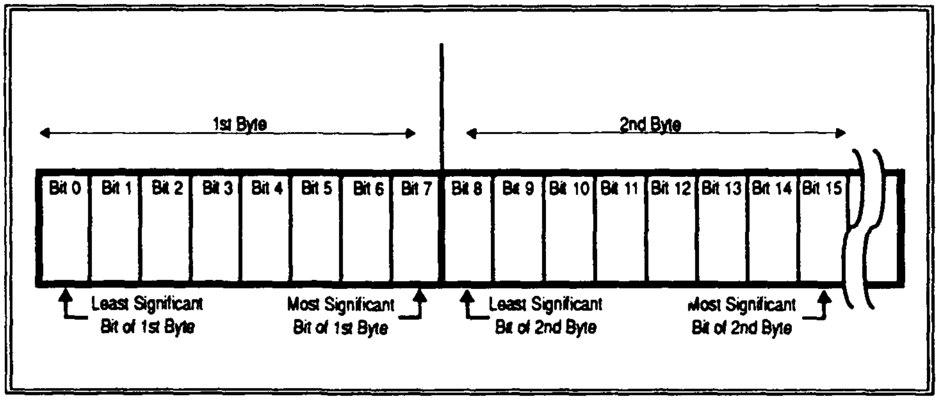

1st Byte
2nd Byte
Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
Bit 8
Bit 9
Bit 10
Bit 11
Bit 12
Bit 13
Bit 14
Bit 15
Least Significant Bit of 1st Byte
Most Significant Bit of 1st Byte
Least Significant Bit of 2nd Byte
Most Significant Bit of 2nd Byte

Figure 12
A bit-vector is a sequence of contiguous bits terminated at a byte boundary.

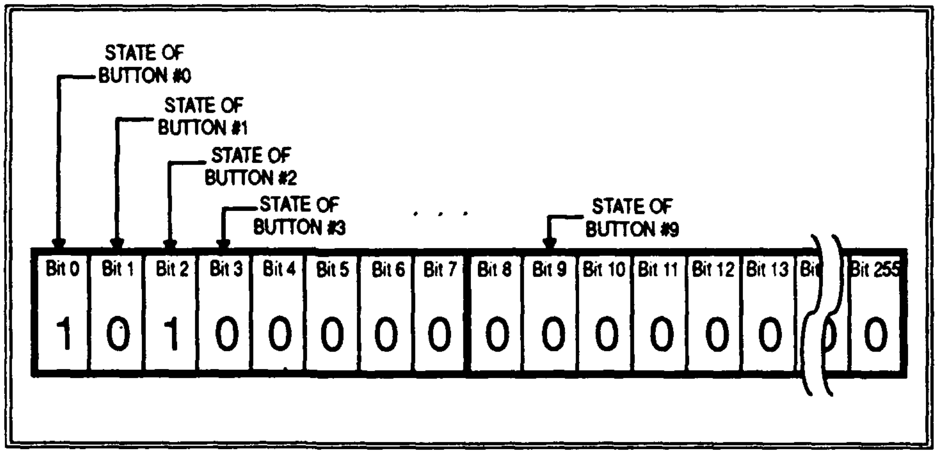


Figure 13
A locator's button vector reports the state of up to 256 logical buttons.

A locator's services can be performed by several peripherals, including mice, touchpanels, graphics tablets, light pens, track balls, joy sticks, and keyboards. Many do not have physical buttons, and some do not have native coordinate systems with stationary, intersecting, perpendicular axes. When used as locators, these peripherals' device handlers must translate their native position and state information into the coordinate system and button vector described above.

For example, most touchpanels have no buttons but can sense the presence or absence of a touch on their surface. When used as a locator, a touchpanel's device handler should use button 0 to report touch activity.

[4] A bit-vector is a string of consecutive bytes whose bits are labeled sequentially from the least significant bit of the first byte to the most significant bit of the last byte. The first bit is labeled 0, the second is labeled 1, etc. Figure 12 illustrates the format of a bit-vector.

Most mice report their position using a native coordinate system whose origin moves with them: Changes in position are reported as a distance from the mouse's last position, not as a distance from a fixed origin. When used as a locator, a mouse's device handler should maintain a fixed coordinate system and map its peripheral's native coordinates into this fixed system. To help guide the implementation of locator device handlers, and to ensure compatibility among different peripherals, Van de Wetering and Thomason (1990) presents the conventions used to implement common peripherals as locators.

Locators are often used to let operators select graphic objects or locations on a display device. When used for this purpose, a peripheral may allow direct or indirect interaction with the screen. Direct interaction occurs if the plane of the peripheral's native coordinate system is physically coincident with the display, and indirect interaction occurs if these planes are not physically coincident. For example, a touchpanel and a light pen both allow direct interaction: Their native coordinate systems are physically coincident with the screen, and selections are made by pointing directly at screen objects or locations. A mouse allows indirect interaction; its coordinate system is usually aligned with the desk top, and selections are made by moving it to indirectly position a screen cursor. This is important because an application must display a cursor to support indirect interactions, while a cursor is not necessary for direct interactions. A locator's device handler reports this attribute using service 8, "Locator Attributes."

The range of a locator's coordinate system and the number of buttons it supports may vary, depending upon each peripheral's physical attributes. For example, a particular touchpanel may only report X and Y coordinates within a range of 0 to 500, and a particular mouse may have only two buttons. These attributes are important because applications often need to align a locator's coordinate system with the coordinate system of another device (e.g., a graphic display) and because some applications require locators with a minimum number of buttons. A locator's device handler reports these physical constraints using service 8, "Locator Attributes."

Table 13 lists all services provided by a logical locator and Table 14 lists all alerts. Table 15 lists all of the status codes returned by locator services.

Table 13
A logical locator provides 11 core services and one extended service.

Name	Service ID	Core/Extended	Page [a]
Initialize Device Handler	0	Core	B-1
Terminate Device Handler	1	Core	B-2
Services Query	2	Core	B-3
Set Alert Mask	3	Core	B-4
Disable All Alerts	4	Core	B-4
Reset	5	Core	B-5
Locator On	6	Core	B-6
Locator Off	7	Core	B-6
Locator Attributes	8	Core	B-7
Locator State	9	Core	B-8
Set Position	10	Core	B-9
Redefine Range	11	Extended	B-10

[a] Pages appear in Appendix B of this publication.

Table 14
A logical locator provides three alerts.

Name	Alert ID	Core/Extended	Page [a]
Movement	0	Core	B-11
Button Pressed	1	Core	B-12
Button Released	2	Core	B-13

[a] Pages appear in Appendix B of this publication.

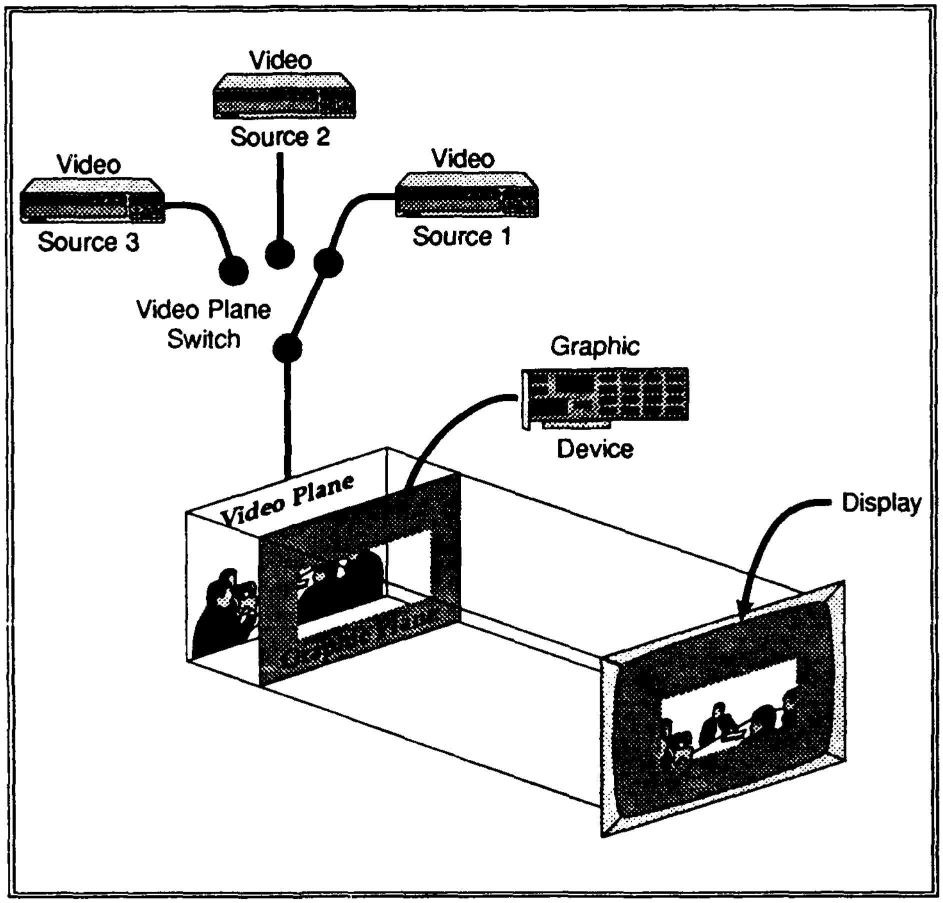
Table 15
A logical locator provides nine error codes.

Description	Error Code
Service successful	0
Unknown error	1
Device not available	2
Device handler busy	3
Locator not responding	4
Locator turned off	5
X-coordinate out of range	6
Y-coordinate out of range	7
Already initialized	8

7.3 Video/Graphics Overlay

A logical overlay device mixes two planes of visual information onto a single display device. Mixing is accomplished by controlling the transparency of the top plane, allowing portions of the bottom plane to show through. Figure 14 illustrates this concept. Imagery on each plane in Figure 14 is controlled by other, independent devices. The top plane, usually controlled by a graphic device, is called the graphic plane. The bottom plane presents imagery from one of several devices (videodisc players, VCRs, etc.) called external video sources. This plane is called the video plane. Service 8, "Select Video Input," is provided to select one source at a time for this plane and is depicted in Figure 14 by the video plane switch. Service 11, "Align Graphic Plane," is used to adjust the two planes' relative alignment so applications can allow operators to calibrate the two displays.

Figure 14
A logical overlay device mixes two planes of visual information onto a single display.



The overlay device's graphic plane is made up of thousands of tiny fixed rectangular regions, called pixels. A graphic device paints pictures on this plane by adjusting the color of each pixel. For example, to paint a red horizontal line on a white background, a graphic device might paint all pixels on the graphic plane white, then color all pixels in the fourth row red. Figure 15 illustrates this concept.

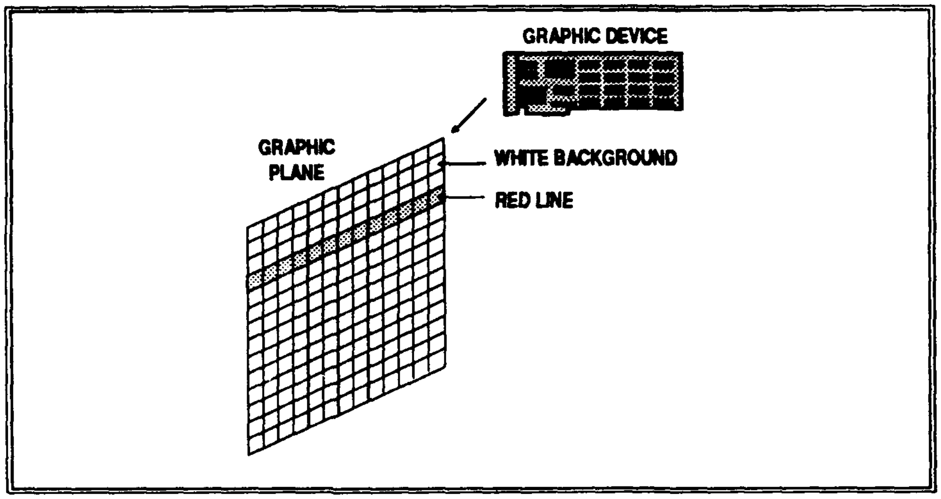


Figure 15
A graphic device paints pictures by adjusting the color of individual pixels.

Graphic devices paint pixels by assigning numbers to them and then using a color palette to assign a hue to each number. Figure 16 demonstrates this technique by presenting a red (1) box with a yellow (2) interior on a blue (0) background. Numbers assigned to pixels in the graphic plane are called logical colors, and can range from 0 to 65535.

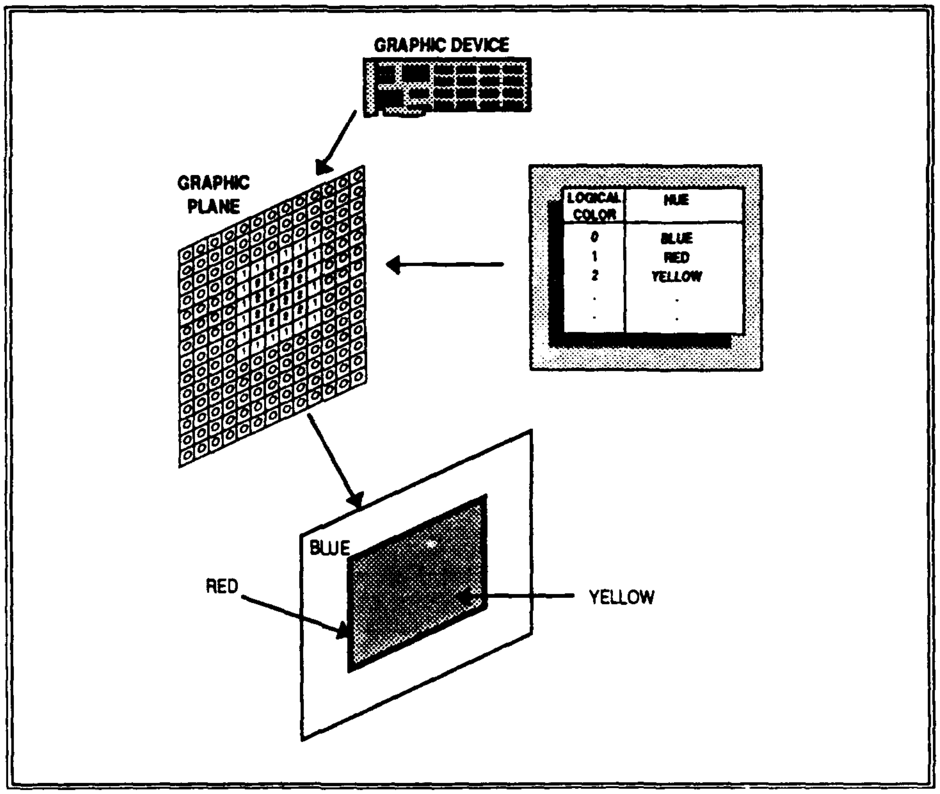
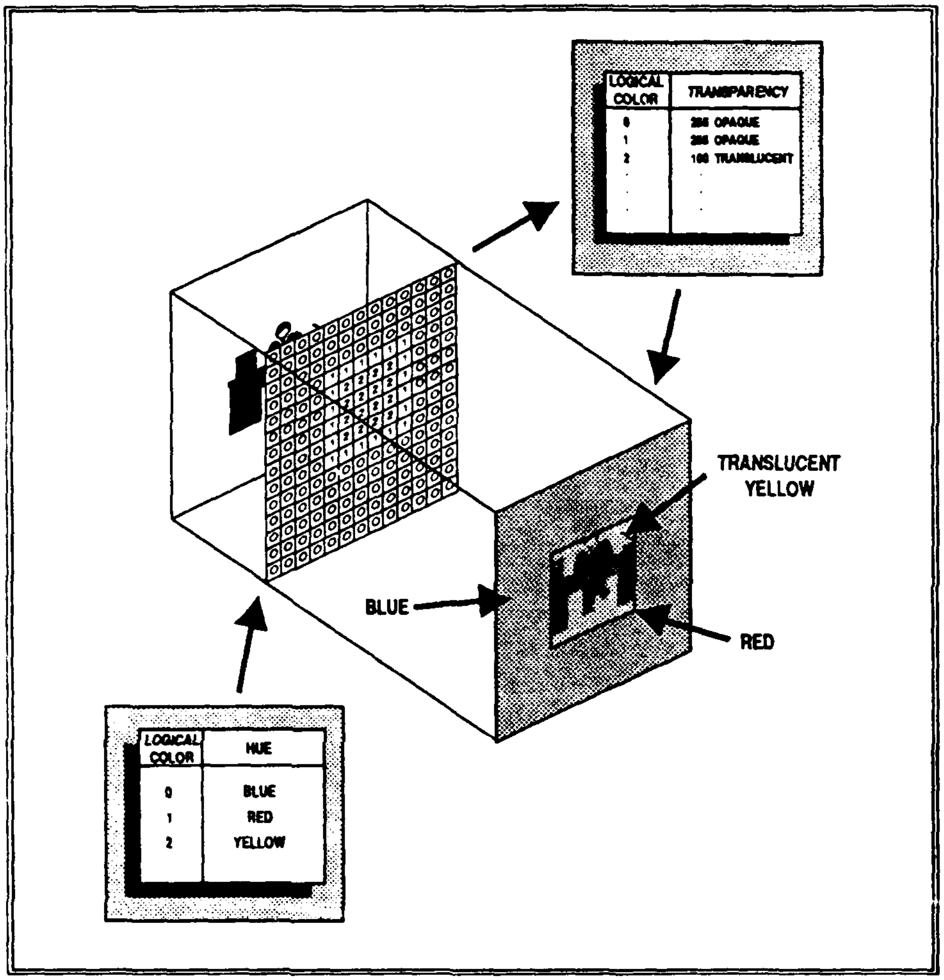


Figure 16
Graphic devices paint pixels by assigning numbers, called logical colors, to them.

Just as a graphic device uses a color palette to assign hues to each logical color, the overlay device uses a transparency palette to assign a transparency to each logical color. Figure 17 illustrates this principle. In this figure, the graphic device has painted a red (1) box with a yellow (2) interior on a blue (0) background. The overlay device's transparency palette has prescribed logical colors 0 and 1 as opaque (255) and logical color 2 as translucent (100). The resulting mixed image presents an opaque blue background with an opaque red box filled with a yellow-tinged (translucent) video image.

Figure 17
A transparency palette assigns a transparency to each logical color.



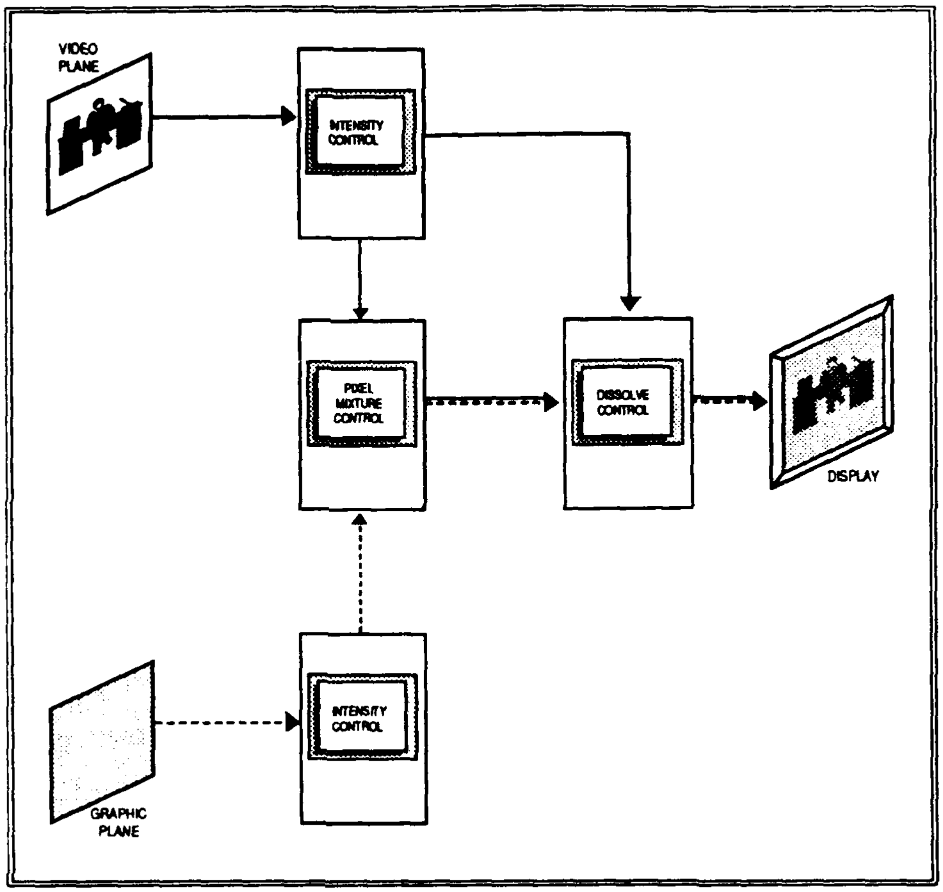
LOGICAL COLOR	TRANSPARENCY
0	255 OPAQUE
1	255 OPAQUE
2	100 TRANSLUCENT
.	.
.	.
.	.

LOGICAL COLOR	HUE
0	BLUE
1	RED
2	YELLOW

A transparency palette has 256 entries, and each entry defines the transparency of a single logical color. Each entry can range from 0 (transparent) to 255 (opaque). Service 6, "Set Colors Transparent/Opaque," is provided to set palette entries to either 0 or 255, and service 13, "Set Colors Translucent," is provided to set translucent tones.

To mix pictures from its two planes onto a single display, an overlay device uses the model illustrated in Figure 18. Lines with arrows in this figure represent pictures flowing from the graphic and video planes (on the left) through various controls (boxes in the middle) to the display (on the right). Dark lines with bold arrows represent video pictures, dotted lines with bold arrows reflect graphic pictures, and double lines (one solid, one dotted) with bold arrows depict an overlaid image (part video, part graphic).

Figure 18
An overlay device uses this model to mix pictures from its two planes on a single display.



The first boxes encountered by both graphic and video pictures in this model are labeled "Intensity Control." As this name suggests, an overlay device can adjust the intensity (or brightness) of pictures from both its video and its graphic planes before mixing them. Services 9, "Set Video Intensity On/Off," and 10, "Set Graphic Intensity On/Off," toggle the intensity of each picture between maximum (on) and minimum (off) brightness. Like adjusting a television's brightness knob, *extended services* 14, "Set Video Intensity," and 15, "Set Graphic Intensity," provide 256 gradations of intensity between 0 (off) and 255 (maximum brightness).

In the middle of Figure 18, both pictures flow through a "Pixel Mixture Control" box. This box scans each graphic pixel and mixes it with video according to its color's transparency. Pixels with opaque colors have no effluent video component, while transparent pixels are mixed with full video (they have no effluent graphic component). Translucent colors are mixed with a relative portion of video.

The pixel mixture control box uses the transparency palette to determine each pixel's video component. However, it can also be directed to ignore the transparency palette and to arbitrarily view all pixels as opaque, using service 7, "Transparancy Palette Enabled/Disabled." An application might use this service to momentarily hide a display's video image without disrupting its arrangement of transparent and translucent regions.

Before reaching the display, Figure 18's pictures pass through a final mixing box labeled "Dissolve Control." This box adds more video to every pixel, using a scaling factor called the dissolve level. Service 16, "Set Dissolve," is provided to regulate the dissolve level between 0 percent and 100 percent. Service 16 is commonly used to vary this scaling factor over a few seconds to produce the effect of dissolving graphics.

Quantitatively, the dissolve level is applied as a scaling factor to each pixel's transparency, as illustrated in Equation 1, to determine the pixel's final display mixture.

Equation 1
Final pixel display mixture

Final Pixel Mixture = (Pixel Transparency) X (Dissolve Level)

In this equation, both the pixel's transparency (as determined by its color's transparency palette entry) and its final mixture range from 255 (opaque, no video) to 0 (transparent, all video). When the dissolve level is 100 percent, it has no effect upon the pixel's appearance. (This is its default level.) When the dissolve level is 0 percent, all pixels become transparent (their final mixture is 0). A display's graphics will disappear as the dissolve level varies from 100 percent to 0 percent, and will reappear as it returns to 100 percent.

Table 16 lists all services provided by a logical overlay device and Table 17 lists all alerts. Table 18 lists all status codes returned by a logical overlay device.

Table 16
A logical overlay device provides 13 core services and four extended services.

Name	Service ID	Core/Extended	Page[a]
Initialize Device Handler	0	Core	C-1
Terminate Device Handler	1	Core	C-2
Services Query	2	Core	C-3
Set Alert Mask	3	Core	C-4
Disable All Alerts	4	Core	C-4
Reset	5	Core	C-5
Set Colors Transparent/Opaque	6	Core	C-6
Transparency Palette Enabled/Disabled	7	Core	C-7
Select Video Input	8	Core	C-7
Set Video Intensity On/Off	9	Core	C-8
Set Graphic Intensity On/Off	10	Core	C-8
Align Graphic Plane	11	Core	C-9
Status Query	12	Core	C-10
Set Colors Translucent	13	Extended	C-12
Set Video Intensity	14	Extended	C-13
Set Graphic Intensity	15	Extended	C-14
Set Dissolve	16	Extended	C-15

[a] Pages appear in Appendix C of this publication.

Table 17
A logical overlay device provides two core alerts.

Name	Alert ID	Core/Extended	Page [a]
Timed Service Completed	0	Core	C-16
Graphic Plane Mode Changed	1	Core	C-16

[a] Pages appear in Appendix C of this publication.

Table 18
A logical overlay device provides 11 error codes.

Description	Error Code
Service successful	0
Unknown error	1
Device not available	2
Device handler busy	3
Hardware not responding	4
Palette bound out of range	5
Invalid transparency palette entry	6
Video source out of range	7
Horizontal shift out of range	8
Vertical shift out of range	9
Already initialized	10

7.4 Audio Management

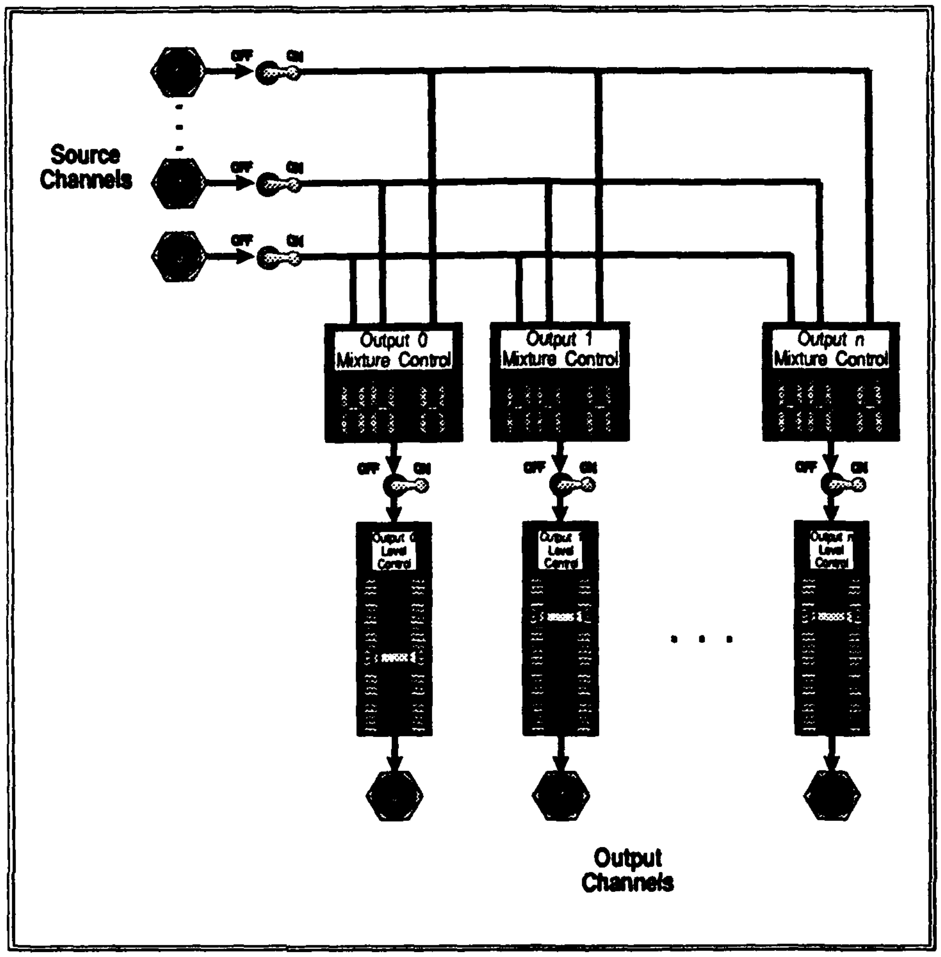
A logical audio management device mixes analog audio signals from several source channels and makes these mixtures available on several output channels. Typical audio sources include videodisc audio, digital audio, and the computer's speaker. Usually two output channels (left and right) are provided for stereo sound. These are typically connected to speakers, headphones, or a stereo amplifier.

To mix sound from several sources, an audio management device uses the model illustrated in Figure 19.

The mixture controls shown in Figure 19 determine the level (loudness) of each source channel before it is mixed. Figure 20 illustrates mixture control for a single output channel. Adjusting the level of each source channel in this figure determines how much sound the channel contributes to the final mixture. Service 8, "Set Source Level," sets the level of a particular source channel to one of 256 gradations of loudness between 0 (silent) and 255 (maximum loudness).

The audio source switches in Figure 19 suggest that an audio management device can turn a source channel on or off without changing its level setting. Service 6, "Audio Source On/Off," turns the selected source channel on or off. When a source channel is turned off its sound does not contribute to the output mixture.

Figure 19
A logical audio management device mixes analog audio signals from several source channels and makes these mixtures available on several output channels.



The output level control boxes shown in Figure 19 suggest that a logical audio management device can control the level of each output channel. Service 9, "Set Output Level," sets the level of a particular output channel to one of 256 gradations of loudness between 0 (silent) and 255 (maximum loudness). Just as with source channels, an audio management device can turn an output channel on or off without changing its level setting. Service 7, "Audio Output On/Off," turns the selected source channel on or off. When a source channel is turned off it is silent.

Table 19 lists all services provided by a logical audio management device, and Table 20 lists all status codes.

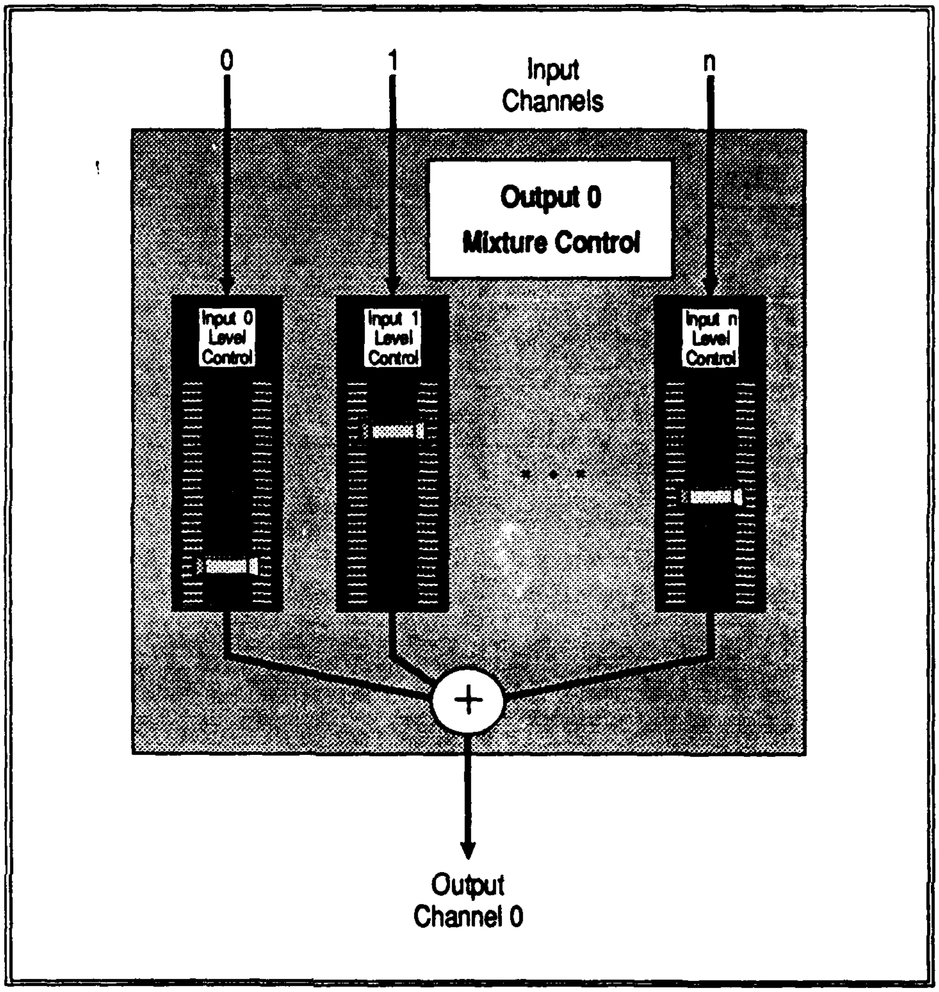


Figure 20
The mixture of source channels can be controlled for each output channel.

Table 19
A logical audio management device provides 12 core services.

Name	Service ID	Core/Extended	Page [a]
Initialize Device Handler	0	Core	D-1
Terminate Device Handler	1	Core	D-2
Services Query	2	Core	D-3
Set Alert Mask	3	Core	D-4
Disable All Alerts	4	Core	D-4
Audio Management Attributes	5	Core	D-5
Audio Source On/Off	6	Core	D-5
Audio Output On/Off	7	Core	D-6
Set Source Level	8	Core	D-7
Set Output Level	9	Core	D-8
Query Source State	10	Core	D-8
Query Output State	11	Core	D-9

[a] Pages appear in Appendix D of this publication.

Table 20
A logical audio management device provides eight error codes.

Description	Error Code
Service successful	0
Unknown error	1
Device not available	2
Device handler busy	3
Hardware not responding	4
Source out of range	5
Output out of range	6
Already initialized	7

8. Glossary

The following list defines terms used in this document. These terms are defined in the context of their use in this document and in the other documents listed in Table 1. The definitions may therefore be more restrictive than, or otherwise differ from, the commonly accepted definitions. Words that are italicized in definitions are themselves defined elsewhere in this list.

Alert A service that asynchronously informs an *application* of device activity.

Alert packet A packet used to communicate an *alert* between *layers* of the *PORTCO architecture.*

Application *Application software.*

Application layer The highest partition in the *PORTCO architecture.* Contains the *DSI toolbox.*

Application software Any software that is part of the *application layer.*

Architecture The organization or structure of a system or of a system component.

Core service A service within a *device class* that must be provided by every compliant *device handler* in the class.

Courseware Software and/or data used to present computer-based instruction.

Device class A collection of services provided by a single *logical device.*

Device handler A *software module* that implements the *DHI* for a single *device class.* Translates requests for service from a *logical device* into instructions that a physical *peripheral* can understand.

Device handler layer The lowest *layer* of the *PORTCO architecture.* Contains *device handlers.*

Device Handler Interface (DHI) A standard collection of services and *protocols* that defines the boundary between the *R&C layer* and the *device handler layer.*

Device Services Interface (DSI) A standard collection of services and *protocols* that defines the boundary between the *application layer* and the *R&C layer.*

DSI toolbox A *software module* (or modules) providing an *interface* between the *application software* and the *R&C layer.*

Extended service A service within a *device class* that may be provided by compliant *device handlers* in the class.

Interface A *software interface.*

Interrupt service routine (ISR) A *software module* invoked by a processor interrupt.

Layer A group of related functions that makes up one level of a *layered architecture.*

Layered architecture A *software* architecture in which components are grouped in a hierarchical arrangement in such a way that each *layer* provides functions and services to adjacent *layers.*

Logical device A conceptual device synthesized by using characteristics of several similar *peripherals.* Specified by a *device class.*

Packet A contiguous block of data used to communicate information between *layers* of the *PORTCO architecture.*

Peripheral device Any computer system hardware component that is distinct from a computer's main processor.

PORTCO architecture A *layered architecture* presenting a standard *interface* between *peripheral devices* and *applications.*

Protocol A set of rules governing the communication between two *software modules.*

Registers The internal registers of the 80x86 processor. These include the data registers (AX, BX, CX, DX), *stack* pointer (SP), base pointer (BP), source index (SI), destination index (DI), code *segment* (CS), data *segment* (DS), *stack segment* (SS), extra *segment* (ES), and flags.

Request packet A packet used to communicate a service request between *layers* of the *PORTCO architecture.*

Routing and configuration layer The middle partition of the *PORTCO architecture.*

R&C layer *Routing and configuration layer.*

Segment A 64-k byte block of memory starting at an address that is evenly divisible by 16.

Software interface The boundary between two or more *software modules*, or a *protocol* that defines how two *software modules* communicate.

Software module A named collection of software instructions and data.

SP The *stack* pointer *register* of the 80x86 processor.

SS The *stack segment register* of the 80x86 processor.

Stack A dynamically shrinking and expanding area of memory in which data items are stored in consecutive order and removed on a last-in, first-out basis. The *stack* is the section of memory used by the CPU to store return addresses from subroutine calls and interrupts.

TSR program An MS-DOS terminate-and-stay-resident program.

9. References

Thomason, B. L., & Van de Wetering, B. L. (April 1990) *The MS-DOS Device Services Interface (TN-90-15).* San Diego: Navy Personnel Research and Development Center.

Van de Wetering, B. L., & Thomason, B. L. (in preparation) *Guidelines for Implementing MS-DOS PORTCO Device Handlers.* San Diego: Navy Personnel Research and Development Center.

Duncan, R. (1986) *Advanced MS-DOS Programming.* Microsoft Press, Redmond, WA.

Thomason, B. L., Van de Wetering, B. L., & Booth, R. G. (March 1990) *A Portable Courseware Architecture (TN-90-11).* San Diego: Navy Personnel Research and Development Center.

APPENDIX A

VIDEODISC PLAYER SERVICES

Name **Initialize Device Handler**

Service ID 0

Description Initializes the device handler and its physical device. Invoked before any other service. (Multiple invocations should not affect the device.) Described in Section 4 of the main body of this document.

Block Format

Field Name	Set By[a]	Byte #	Format[b]	Possible Values	Notes
Service ID	RC	0-1	16-bit uns. int.	0	
Block Length	RC	2-3	16-bit uns. int.	19	
Status	DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 19 = Already initialized	(1)
Init String Address	RC	5-8	32-bit address		(2)
R&C Alert Address	RC	9-12	32-bit address		(3)
Available Memory Bytes	RC	13-16	32-bit uns. int.		(4)
Next Segment	DH	17-18	16-bit address		(5)

[a] Throughout Appendix A, this column indicates which program(s) set the specified field: A = application; RC = R&C program; DH = device handler.

[b] Throughout Appendix A, the abbreviations used in this column are: "uns. int." = unsigned integer; "2's comp. int." = two's complement integer.

Notes

1. The device handler should take no action when this service is invoked multiple times.
2. The syntax and semantics of a device handler's initialization string are dictated by the device handler developer. The initialization string appears in the DSI configuration file (see Thomason and Van de Wetering (1990) in the main body of this document).
3. Address of R&C program entry point that routes alert packets to the application.
4. Unused memory that is contiguous to the device handler and available for its use. (Device handler must reserve memory for any alert packets it will use.)
5. Address set by device handler where R&C program may load another device handler without interfering with subject handler. See Section 3 in the main body of this document.

Name **Terminate Device Handler**

Service ID 1

Description Prepare the physical peripheral to be powered off. Restore any system interrupt vectors altered during device handler initialization and operation. This service immediately overrides any other command.

Block Format

Field Name	**Set By**	**Byte #**	**Format**	**Possible Values**
Service ID	RC	0-1	16-bit uns. int.	1
Block Length	RC	2-3	16-bit uns. int.	5
Status	DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error

Name **Services Query**

Service ID 2

Description Lists all services and alerts provided by the target device. Returns two bit-vectors, one describing services and one describing alerts. The N'th bit in each vector reflects support for the N'th service or alert. Allows an application to determine which extended services are supported by a particular device.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	2	
Block Length	A	2-3	16-bit uns. int.	15	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy	
Request Vector Byte Length	DH	5	8-bit uns. int.		
Request Vector Address	DH	6-9	32-bit address to bit-vector		(1)
Alert Vector Byte Length	DH	10	8-bit uns. int.		
Alert Vector Address	DH	11-14	32-bit address to bit-vector		(1)

Note

1. Bits set to 1 indicate a supported service. Bits set to 0 reflect an unsupported service.

Name **Set Alert Mask**

Service ID 3

Description Enables and disables alerts based on the contents of a bit-vector, called the alert mask. This mask has the same format as service 2's alert vector: the N'th bit in the mask corresponds with the N'th alert. If the bit is 1, the device handler will issue the alert. If the bit is 0, the device handler will not issue the alert. All alerts are disabled by default.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	3
Block Length	A	2-3	16-bit uns. int.	10
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy
Alert Mask Byte Length	A	5	8-bit uns. int.	
Alert Mask Address	A	6-9	32-bit address to bit-vector	

Name **Disable All Alerts**

Service ID 4

Description Disables all alerts.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	4
Block Length	A	2-3	16-bit uns. int.	5
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy

Name **Player Status**

Service ID 5

Description Returns current videodisc player status.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	5
Block Length	A	2-3	16-bit uns. int.	12
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding
Videodisc Status	DH	5	8-bit uns. int.	0 = Ready 1 = Ejected 2 = No videodisc in player 3 = Videodisc parked
Audio Channel 1	DH	6	8-bit uns. int.	0 = Channel 1 off 1 = Channel 1 on
Audio Channel 2	DH	7	8-bit uns. int.	0 = Channel 2 off 1 = Channel 2 on
Frame Display	DH	8	8-bit uns. int.	0 = Frame numbers off 1 = Frame numbers on
Chapter Display	DH	9	8-bit uns. int.	0 = Chapter numbers off 1 = Chapter numbers on
Video On/Off	DH	10	8-bit uns. int.	0 = Video off 1 = Video on
Current Motion	DH	11	8-bit 2's comp. int.	-4 = Scan reverse -3 = Fast reverse -2 = Slow reverse -1 = Normal reverse 0 = Still 1 = Normal forward 2 = Slow forward 3 = Fast forward 4 = Scan forward

Name **Videodisc Status**

Service ID 6

Description Returns information about the current videodisc.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	6	
Block Length	A	2-3	16-bit uns. int.	15	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding	
Videodisc Status	DH	5	8-bit uns. int.	0 = Ready 1 = Ejected 2 = No videodisc in player 3 = Videodisc parked	
Type	DH	6	8-bit uns. int.	0 = CAV 1 = CLV 2 = Other	
Format	DH	7	8-bit uns. int.	0 = NTSC 1 = PAL 2 = Other	
Size	DH	8	8-bit uns. int.	0 = Unknown >0 = Diameter in centimeters	
Chapter Coding	DH	9	8-bit uns. int.	0 = Unavailable 1 = Available	
Side Number	DH	10	8-bit uns. int.	1 = Side one active 2 = Side two active 3 = Unknown	
First Frame Number	DH	11-12	16-bit uns. int.		(1)
Last Frame Number	DH	13-14	16-bit uns. int.		(1)

Notes 1. Integer between 1 and 54,000.

Name **Position Request**

Service ID 7

Description Returns the current videodisc frame number.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	7	
Block Length	A	2-3	16-bit uns. int.	7	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 21 = Player spun down	
Frame Number	DH	5-6	16 bit uns. int.		(1)

Note 1. If the videodisc is currently playing motion video, the position may not be accurate because of the time required for communication with the player.

Name **Spin Up/Down**

Service ID 8

Description Spins the videodisc up or down. After the videodisc is spun down, the player will be unable to display an image or play an audio track until it is spun up again. After the videodisc is spun up, the player is placed in still mode on the videodisc's first readable frame. Only services 0 through 6 can be successfully invoked when the player is not spun up. If this service attempts to place the videodisc player in the state it is already in, the service has no effect.

Block Format

Field Name	**Set By**	**Byte #**	**Format**	**Possible Values**	**Notes**
Service ID	A	0-1	16-bit uns. int.	8	
Block Length	A	2-3	16-bit uns. int.	7	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 7 = Ejected 8 = No videodisc in player 21 = Player spun down	
Sync/ Async	A	5	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(1)
Spin Up/Down	A	6	8-bit uns. int.	0 = Spin videodisc down 1 = Spin videodisc up	

Note

1. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," is issued when the service is complete or has failed. If any other service is invoked during an asynchronous invocation of service 8, "Spin Up/Down," a "player busy" status will be returned.

Name **Eject**

Service ID 9

Description Spins down the videodisc and opens the videodisc player door so the videodisc may be removed. When executed asynchronously, only services 0 through 6 can be successfully invoked before this service completes. If any other service is invoked before an asynchronous invocation of service 9, "Eject," completes, a "player busy" status will be returned.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	9	
Block Length	A	2-3	16-bit uns. int.	6	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 7 = Ejected 20 = Must open with front panel	
Sync/ Async	A	5	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(1)

Note

1. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," is issued when the service is complete or has failed.

Name **Frame Search**

Service ID 10

Description Searches to the specified frame. Upon reaching the specified frame, the videodisc player enters still mode. When executed asynchronously, only services 1 through 6 can be successfully invoked before this service completes. If any other service is invoked before an asynchronous invocation of service 10, "Frame Search," completes, a "player busy" status will be returned. This service does not affect service 14, "Video On/Off." However, video output may be blanked while the player is searching, depending on the player and the search distance.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	10	
Block Length	A	2-3	16-bit uns. int.	8	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 9 = Invalid frame number 21 = Player spun down	
Sync/ Async	A	5	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(1)
Target Frame	A	6-7	16-bit uns. int.	Positive integer between 1 and 54,000	

Note

1. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," is issued when the service is complete or has failed.

Name **Jump**

Service ID 11

Description Causes a jump, without video blanking, to a new frame that is within 99 frames in either direction of the current frame. When executed asynchronously, only services 1 through 6 can be successfully invoked before this service completes. If any other service is invoked before an asynchronous invocation of service 11, "Jump," completes, a "player busy" status will be returned.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	11	
Block Length	A	2-3	16-bit uns. int.	7	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 11 = Invalid jump offset 13 = Destination unreachable 21 = Player spun down	
Sync/ Async	A	5	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(1)
Jump Offset Frames/ Direction	A	6	8-bit 2's comp. int.	-1 to -99 = Jump N frames backward 0 = No jump 1 to 99 = Jump N frames forward	

Note

1. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," is issued when the service is complete or has failed.

Name **Still**

Service ID 12

Description Immediately overrides asynchronous execution of service 22, "Chapter Play," or service 23, "Motion," and causes the videodisc player to enter still mode at the current frame. Attempts to execute this command during any other asynchronous commands will result in "player busy" status.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	12
Block Length	A	2-3	16-bit uns. int.	5
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 21 = Player spun down

Name **Chapter Search**

Service ID 13

Description Searches to the specified chapter. On reaching the specified chapter, the videodisc player enters still mode. When executed asynchronously, only services 1 through 6 can be successfully invoked before this service completes. If any other service is invoked before an asynchronous invocation of service 13, "Chapter Search," completes, a "player busy" status will be returned. Execution of this service does not affect service 14, "Video On/Off." However, video output may be blanked on searching to the specified chapter, depending on the player and the search distance.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	13	
Block Length	A	2-3	16-bit uns. int.	8	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 10 = Invalid chapter number 12 = Chapters not available 21 = Player spun down	
Sync/ Async	A	5	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(1)
Target Chapter	A	6-7	16-bit uns. int.		

Note

1. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," is issued when the service is complete or has failed.

Name **Video On/Off**

Service ID 14

Description Enables and disables video output from the videodisc player. Other videodisc player services do not change this state. When this service disables the video signal, only another invocation of this service will enable it.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	14
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 7 = Ejected 21 = Player spun down
Video On/Off	A	5	8-bit uns. int.	0 = Turn video signal off 1 = Turn video signal on

Name **Audio1 On/Off**

Service ID 15

Description Enables and disables output from audio channel 1 at the videodisc player. The state of audio channel 1 does not affect the state of audio channel 2.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	15
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 7 = Ejected 21 = Player spun down
Audio1 On/Off	A	5	8-bit uns. int.	0 = Turn audio channel 1 off 1 = Turn audio channel 1 on

Name **Audio2 On/Off**

Service ID 16

Description Enables and disables output from audio channel 2 at the videodisc player. The state of audio channel 2 does not affect the state of audio channel 1.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	16
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 7 = Ejected 21 = Player spun down
Audio2 On/Off	A	5	8-bit uns. int.	0 = Turn audio channel 2 off 1 = Turn audio channel 2 on

Name **Set Stop**

Service ID 17

Description Specifies a frame at which the videodisc player will enter still mode in any current or future motion commands. This command used in conjunction with the motion command constitutes a bounded play. Alert 1, "Frame Arrival," is issued when the specified frame is reached. Once reached, exceeded, or replaced by a subsequent invocation of the service, the specified frame number has no significance. If possible, the player stops at the exact frame specified. Under conditions that make this impossible (i.e., fast motion), error is kept to a minimum.

Block Format

Field	**Set**	**Byte**		**Possible**
Name	**By**	**#**	**Format**	**Values**
Service ID	A	0-1	16-bit uns. int.	17
Block Length	A	2-3	16-bit uns. int.	7
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 9 = Invalid frame number 21 = Player spun down
Target Frame	A	5-6	16-bit uns. int.	

Name **Set Inform Flag**

Service ID 18

Description Specifies a frame that, when reached, will cause the device handler to issue alert 1, "Frame Arrival."

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	18
Block Length	A	2-3	16-bit uns. int.	7
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 9 = Invalid frame number 21 = Player spun down
Target Frame	A	5-6	16-bit uns. int.	

Name **Remote Control On/Off**

Service ID 19

Description Enables and disables the videodisc player's remote and front-panel controls. When the remote control is disabled, the videodisc player is under computer control. When enabled, the player can be controlled by either the remote unit or the computer.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	19
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 21 = Player spun down
On/Off	A	5	8-bit uns. int.	0 = Disable remote control 1 = Enable remote control

Name **Frame Display On/Off**

Service ID 20

Description Enables and disables display of the current frame number as part of the video picture.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	20
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 7 = Ejected 21 = Player spun down

On/Off	A	5	8-bit uns. int.	0 = Turn frame # display off
				1 = Turn frame # display on

Name **Chapter Display On/Off**

Service ID 21

Description Enables and disables display of the current chapter number as part of the video picture.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	21
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 7 = Ejected 12 = Chapters not available 21 = Player spun down

On/Off	A	5	8-bit uns. int.	0 = Turn chapter # display off
				1 = Turn chapter # display on

Name **Chapter Play**

Service ID 22

Description Searches to the specified chapter and begins to play forward at normal speed until the end of the chapter is reached. At the end of the chapter, the videodisc player enters still mode and video output is turned off. During asynchronous operation, all other services operate as specified. Execution of this service does not affect service 14, "Video On/Off'; however, video output may be blanked while searching to the beginning of the specified chapter, depending on the player and the search distance.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	22	
Block Length	A	2-3	16-bit uns. int.	8	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 10 = Invalid chapter number 12 = Chapters not available 18 = Interrupted before completion 21 = Player spun down	
Sync/ Async	A	5	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(1)
Target Chapter	A	6-7	16-bit uns. int.		

Note

1. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," will be issued when the service has been completed or has failed. If another service request interrupts asynchronous performance, alert 0 will return a status of 18, "Interrupted Before Completion."

Name **Motion**

Service ID 23

Description Causes the videodisc player to begin to play in the specified direction at the specified speed from the current frame. Returns immediately after initiating motion, and is thus inherently asynchronous. All other services operate as specified during motion. Alert 1, "Frame Arrival," will be issued when motion reaches the beginning or end of the videodisc.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	23
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy 6 = Videodisc parked 7 = Ejected 8 = No videodisc in player 14 = Invalid speed selection 15 = Player already at first frame 16 = Player already at last frame 21 = Player spun down
Speed/ Direction	A	5	8-bit 2's comp. int.	-4 = Scan reverse -3 = Fast reverse -2 = Slow reverse -1 = Normal reverse 0 = Still 1 = Normal forward 2 = Slow forward 3 = Fast forward 4 = Scan forward

Name **Service Status Query**

Service ID 24

Description Requests the completion status of the most recently solicited service.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	24	
Block Length	A	2-3	16-bit uns. int.	8	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Player not responding 5 = Player busy	
ID of Reported Service	DH	5-6	16-bit uns. int.		
Status of Reported Service	DH	7	8-bit uns. int.		(1)

Note 1. Status values are listed in the specification for each request.

Name **Service Complete**

Alert ID 0

Description Issued when an asynchronously executed service has been successfully completed or has failed.

Block Format

Field	Set	Byte		Possible	
Name	By	#	Format	Values	Notes
Alert ID	DH	0-1	16-bit uns. int.	0	
Block Length	DH	2-3	16-bit uns. int.	7	
ID of Reported Service	DH	4-5	16-bit uns. int.	6 = Spin up/down 7 = Eject 8 = Frame search 9 = Jump 11 = Chapter search 20 = Chapter play	
Status of Reported Service	DH	6	8-bit uns. int.		(1)

Note 1. Status values are listed in the specification for each request.

Name **Frame Arrival**

Alert ID 1

Description Issued when the frame set by the "Set Stop" or "Set Inform Flag" service has been reached or when service 23, "Motion," reaches the beginning or end of the videodisc.

Block Format

Field	Set	Byte		Possible
Name	By	#	Format	Values
Alert ID	DH	0-1	16-bit uns. int.	1
Block Length	DH	2-3	16-bit uns. int.	6
Service ID	DH	4-5	16-bit uns. int.	17 = Set stop 18 = Set inform flag 23 = Motion

APPENDIX B

LOCATOR SERVICES

Name **Initialize Device Handler**

Service ID 0

Description Initializes the device handler and its physical device. Invoked before any other service. (Multiple invocations should not affect the device.) Described in Section 4 of the main body of this document.

Block Format

Field Name	Set By[a]	Byte #	Format[b]	Possible Values	Notes
Service ID	RC	0-1	16-bit uns. int.	0	
Block Length	RC	2-3	16-bit uns. int.	19	
Status	DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 8 = Already initialized	(1)
Init String Address	RC	5-8	32-bit address		(2)
R&C Alert Address	RC	9-12	32-bit address		(3)
Available Memory Bytes	RC	13-16	32-bit uns. int.		(4)
Next Segment	DH	17-18	16-bit address		(5)

[a] Throughout Appendix B, this column indicates which program(s) set the specified field: A = application; RC = R&C program; DH = device handler.

[b] Throughout Appendix B, the abbreviations used in this column are: "uns. int." = unsigned integer; "2's comp. int." = two's complement integer.

Notes

1. The device handler should take no action when this service is invoked multiple times.
2. The syntax and semantics of a device handler's initialization string are dictated by the device handler developer. The initialization string appears in the DSI configuration file (see Thomason and Van de Wetering (1990) in the main body of this document).
3. Address of R&C program entry point that routes alert packets to the application.
4. Unused memory that is contiguous to the device handler and available for its use. (Device handler must reserve memory for any alert packets it will use.)
5. Address set by device handler where R&C program may load another device handler without interfering with subject handler. See Section 3 in the main body of this document.

Name **Terminate Device Handler**

Service ID 1

Description Prepare the physical peripheral to be powered off. Restore any system interrupt vectors altered during device handler initialization and operation. This service immediately overrides any other command.

Block Format

Field	Set	Byte		Possible
Name	By	#	Format	Values
Service ID	RC	0-1	16-bit uns. int.	1
Block Length	RC	2-3	16-bit uns. int.	5
Status	DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error

Name **Services Query**

Service ID 2

Description Lists all services and alerts provided by the target device. Returns two bit-vectors, one describing services and one describing alerts. The N'th bit in each vector reflects support for the N'th service or alert. Allows an application to determine which extended services are supported by a particular device.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	2	
Block Length	A	2-3	16-bit uns. int.	15	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy	
Request Vector Byte Length	DH	5	8-bit uns. int.		
Request Vector Address	DH	6-9	32-bit address to bit-vector		(1)
Alert Vector Byte Length	DH	10	8-bit uns. int.		
Alert Vector Address	DH	11-14	32-bit address to bit-vector		(1)

Note 1. Bits set to 1 indicate a supported service. Bits set to 0 reflect an unsupported service.

Name **Set Alert Mask**

Service ID 3

Description Enables and disables alerts based on the contents of a bit-vector, called the alert mask. This mask has the same format as service 2's alert vector: The N'th bit in the mask corresponds with the N'th alert. If the bit is 1, the device handler will issue the alert. If the bit is 0, the device handler will not issue the alert. All alerts are disabled by default.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	3
Block Length	A	2-3	16-bit uns. int.	10
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy
Alert Mask Byte Length	A	5	8-bit uns. int.	Length of mask in bytes
Alert Mask Address	A	6-9	32-bit address to bit-vector	

Name **Disable All Alerts**

Service ID 4

Description Disables all alerts.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	4
Block Length	A	2-3	16-bit uns. int.	5
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy

Name **Reset**

Service ID 5

Description Causes the locator's device handler to establish communications with its peripheral, perform any hardware-specific initialization, set the current location of the locator to (0, 0), and turn the locator on so that it is ready to respond to subsequent service requests. For example, a serial touchpanel handler might reset its communications channel, revector the serial port interrupt if necessary, and reset the touchpanel.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	5	
Block Length	A	2-3	16-bit uns. int.	5	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Locator not responding	(1)

Note

1. Individual manufacturers may define additional error codes to indicate malfunctions specific to their hardware.

Name **Locator On**

Service ID 6

Description Causes the locator's device handler to process all subsequent services, to respond to activity from the peripheral by asynchronously recording position changes, and to issue all alerts that have not been masked.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	6
Block Length	A	2-3	16-bit uns. int.	5
Status	RC/DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Locator not responding

Name **Locator Off**

Service ID 7

Description Causes the locator's device handler to ignore subsequent activity from the peripheral, to respond to all service requests except service 6, "Locator On," and service 8, "Locator Attributes," with an error code, and to stop issuing alerts regardless of the alert mask. This service should be invoked when an application wants to ignore locator input for an extended time. Turning the locator off may improve system performance by minimizing device handler background processing.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	7
Block Length	A	2-3	16-bit uns. int.	5
Status	RC/DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Locator not responding

Name **Locator Attribıutes**

Service ID 8

Description Provides information about the locator's coordinate system and number of buttons.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	8	
Block Length	A	2-3	16-bit uns. int.	19	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Locator not responding	
Button Config- uration	DH	5	8-bit uns. int.	Number of locator buttons	
Maximum X-Value	DH	6-7	16-bit 2's comp. int.		
Maximum Y-Value	DH	8-9	16-bit 2's comp. int.		
Minimum X-Value	DH	10-11	16-bit 2's comp. int.		
Minimum Y-Value	DH	12-13	16-bit 2's comp. int.		
X-Aspect	DH	14-15	16-bit 2's comp. int.		(1)
Y-Aspect	DH	16-17	16-bit 2's comp. int.		(1)
Direct/ Indirect Interaction	DH	18	8-bit uns. int.	0 = Direct interaction 1 = Indirect interaction 2 = Interaction type unknown	

Note

1. Used to calculate the locator's aspect ratio. This ratio is: X-aspect/Y aspect. The aspect ratio indicates the physical size of one unit on the X-axis relative to the physical size of one unit on the Y-axis. For example, an aspect ratio of 1 indicates that the units on the X-axis represent the same physical distance as the units on the Y-axis. An aspect ratio of 2 indicates that the units on the X-axis represent twice the physical distance of the units on the Y-axis.

Name **Locator State**

Service ID 9

Description Provides information about the current locator state. Returns the most recent locator position and the state of all buttons.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	9	
Block Length	A	2-3	16-bit uns. int.	41	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Locator not responding 5 = Locator turned off	
X-Coordinate	DH	5-6	16-bit 2's comp. int.		
Y-Coordinate	DH	7-8	16-bit 2's comp. int.		
Button Vector	DH	9-40			(1)

Note

1. This field is a bit-vector in which the N'th bit specifies the state of the N'th logical button (see Figure 13 in the main body of this document). If a bit's value is 1, its logical button is pressed; otherwise it is released.

Name **Set Position**

Service ID 10

Description Set the current locator position to coordinates passed in the request block. Allows applications to set a new reference point for peripherals that return only movement information (e.g., mouse, keypad, etc.).

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	10
Block Length	A	2-3	16-bit uns. int.	9
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Locator not responding 5 = Locator turned off 6 = X-coordinate out of range 7 = Y-coordinate out of range
New X-Coordinate	A	5-6	16-bit 2's comp. int.	
New Y-Coordinate	A	7-8	16-bit 2's comp. int.	

Name **Redefine Range**

Service ID 11

Description Changes the boundaries of the locator's coordinate system.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	11
Block Length	A	2-3	16-bit uns. int.	13
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Locator not responding 5 = Locator turned off
Maximum X-Value	A	5-6	16-bit 2's comp. int.	
Maximum Y-Value	A	7-8	16-bit 2's comp. int.	
Minimum X-Value	A	9-10	16-bit 2's comp. int.	
Minimum Y-Value	A	11-12	16-bit 2's comp. int.	

Name **Movement**

Alert ID 0

Description Issued whenever the locator reports a change in position.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Alert ID	DH	0-1	16-bit uns. int.	0	
Block Length	DH	2-3	16-bit uns. int.	40	
X-Coordinate	DH	4-5	16-bit 2's comp. int.		
Y-Coordinate	DH	6-7	16-bit 2's comp. int.		
Button Vector	DH	8-39			(1)

Note

1. This field is a bit-vector in which the N'th bit specifies the state of the N'th logical button (see Figure 13 in the main body of this document). If a bit's value is 1, its logical button is pressed; otherwise it is released.

Name **Button Pressed**

Alert ID 1

Description Issued whenever a locator button changes from released to pressed.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Alert ID	DH	0-1	16-bit uns. int.	1	
Block Length	DH	2-3	16-bit uns. int.	41	
Active Button	DH	4	8-bit uns. int.		
X- Coordinate	DH	5-6	16-bit 2's comp. int.		
Y- Coordinate	DH	7-8	16-bit 2's comp. int.		
Button Vector	DH	9-40			(1)

Note

1. This field is a bit-vector in which the N'th bit specifies the state of the N'th logical button (see Figure 13 in the main body of this document). If a bit's value is 1, its logical button is pressed; otherwise it is released.

Name **Button Released**

Alert ID 2

Description Issued whenever a locator button changes from pressed to released.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Alert ID	DH	0-1	16-bit uns. int.	2	
Block Length	DH	2-3	16-bit uns. int.	41	
Active Button	DH	4	8-bit uns. int.		
X-Coordinate	DH	5-6	16-bit 2's comp. int.		
Y-Coordinate	DH	7-8	16-bit 2's comp. int.		
Button Vector	DH	9-40			(1)

Note

1. This field is a bit-vector in which the N'th bit specifies the state of the N'th logical button (see Figure 13 in the main body of this document). If a bit's value is 1, its logical button is pressed; otherwise it is released.

APPENDIX C

VIDEO/GRAPHICS OVERLAY SERVICES

Name **Initialize Device Handler**

Service ID 0

Description Initializes the device handler and its physical device. Invoked before any other service. (Multiple invocations should not affect the device.) Described in Section 4 of the main body of this document.

Block Format

Field Name	Set By[a]	Byte #	Format[b]	Possible Values	Notes
Service ID	RC	0-1	16-bit uns. int.	0	
Block Length	RC	2-3	16-bit uns. int.	19	
Status	DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 10 = Already initialized	(1)
Init String Address	RC	5-8	32-bit address		(2)
R&C Alert Address	RC	9-12	32-bit address		(3)
Available Memory Bytes	RC	13-16	32-bit uns. int.		(4)
Next Segment	DH	17-18	16-bit address		(5)

[a] Throughout Appendix C, this column indicates which program(s) set the specified field: A = application; RC = R&C program; DH = device handler.
[b] Throughout Appendix C, the abbreviations used in this column are: "uns. int." = unsigned integer; "2's comp. int." = two's complement integer.

Notes

1. The device handler should take no action when this service is invoked multiple times.

2. The syntax and semantics of a device handler's initialization string are dictated by the device handler developer. The initialization string appears in the DSI configuration file (see Thomason and Van de Wetering (1990) in the main body of this document).

3. Address of R&C program entry point that routes alert packets to the application.

4. Unused memory that is contiguous to the device handler and available for its use.

5. Set by device handler. Address where the R&C program may load another device handler without interfering with subject handler. See Section 3 in the main body of this document.

Name **Terminate Device Handler**

Service ID 1

Description Prepare the physical peripheral to be powered off. Restore any system interrupt vectors altered during device handler initialization and operation. This service immediately overrides any other command.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	RC	0-1	16-bit uns. int.	1
Block Length	RC	2-3	16-bit uns. int.	5
Status	DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error

Name **Services Query**

Service ID 2

Description Lists all services and alerts provided by the target device. Returns two bit-vectors, one describing services and one describing alerts. The N'th bit in each vector reflects support for the N'th service or alert. Allows an application to determine which extended services are supported by a particular device.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	2	
Block Length	A	2-3	16-bit uns. int.	15	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy	
Request Vector Byte Length	DH	5	8-bit uns. int.		
Request Vector Address	DH	6-9	32-bit address to bit-vector		(1)
Alert Vector Byte Length	DH	10	8-bit uns. int.		
Alert Vector Address	DH	11-14	32-bit address to bit-vector		(1)

Note 1. Bits set to 1 indicate a supported service. Bits set to 0 reflect an unsupported service.

Name **Set Alert Mask**

Service ID 3

Description Enables and disables alerts based on the contents of a bit-vector, called the alert mask. This mask has the same format as service 2's alert vector: The N'th bit in the mask corresponds with the N'th alert. If the bit is 1, the device handler will issue the alert. If the bit is 0, the device handler will not issue the alert. All alerts are disabled by default.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	3
Block Length	A	2-3	16-bit uns. int.	10
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy
Alert Mask Byte Length	A	5	8-bit uns. int.	
Alert Mask Address	A	6-9	32-bit address to bit-vector	

Name **Disable All Alerts**

Service ID 4

Description Disables all alerts.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	4
Block Length	A	2-3	16-bit uns. int.	5
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy

Name	**Reset**
Service ID	5
Description	Causes the overlay device's device handler to establish communications with its peripheral and to perform any hardware-specific initialization. After successful completion of this service, the graphic plane is turned on, the video plane is turned off, all logical colors are opaque, the current video source is source 0, and transparency is disabled.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	5	
Block Length	A	2-3	16-bit uns. int.	5	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown Error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding	(1)

Note

1. Individual manufacturers may define additional error codes to indicate malfunctions specific to their hardware.

Name **Set Colors Transparent/Opaque**

Service ID 6

Description Turns transparency on or off for logical colors on the graphic plane by changing entries in the transparency palette.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	6	
Block Length	A	2-3	16-bit uns. int.	13	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 5 = Palette bound out of range 6 = Invalid transparency palette entry	
First Color	A	5-6	16-bit uns. int.		(1)
Last Color	A	7-8	16-bit uns. int.		(1)
Palette Address	A	9-12	32-bit address	0 = Transparent 255 = Opaque	(2)

Notes

1. Specifies the upper and lower bounds of a range of logical colors whose transparency palette entries will be changed by this service.
2. The address of an array of new transparency palette entries for the range of logical colors specified by the previous two fields. Each element of this array must be an 8-bit, unsigned integer whose value is either 0 or 255. The number of entries the array should contain equals Last Color - First Color + 1.

Name **Transparency Palette Enabled/Disabled**

Service ID 7

Description Enables and disables the transparency palette. When disabled, all colors are considered opaque.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	7
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding
Enabled/ Disabled	A	5	8-bit uns. int.	0 = Disable transparency 1 = Enable transparency

Name **Select Video Input**

Service ID 8

Description Selects the video source to be displayed on the video plane.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	8	
Block Length	A	2-3	16-bit uns. int.	6	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 7 = Video source out of range	
Video Source	A	5	8-bit uns. int.		

Name **Set Video Intensity On/Off**

Service ID 9

Description Sets the video intensity to maximum (on) or minimum (off).

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	9
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding
Video On/Off	A	5	8-bit uns. int.	255 = On 0 = Off

Name **Set Graphic Intensity On/Off**

Service ID 10

Description Sets the graphic intensity to maximum (on) or minimum (off).

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	10
Block Length	A	2-3	16-bit uns. int.	6
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding
Graphic On/Off	A	5	8-bit uns. int.	255 = On 0 = Off

Name **Align Graphic Plane**

Service ID 11

Description Shifts the position of the graphic plane relative to the video plane.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	11	
Block Length	A	2-3	16-bit uns. int.	9	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 8 = Horizontal shift out of range 9 = Vertical shift out of range	
Horizontal Shift	A	5-6	16-bit 2's comp. int.		(1)
Vertical Shift	A	7-8	16-bit 2's comp. int.		(2)

Notes

1. Specifies the number of pixels to shift the graphic plane horizontally relative to the video plane from its current position. A negative value specifies a shift to the left and a positive value specifies a shift to the right.
2. Specifies the number of pixels to shift the graphic plane vertically relative to the video plane from its current position. A negative value specifies a shift down and a positive value specifies a shift up.

Name **Status Query**

Service ID 12

Description Provides information about the current state of the overlay device.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	20	
Block Length	A	2-3	16-bit uns. int.	21	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 5 = Palette bound out of range	
Palette Size	DH	5-6	16-bit uns. int.		(1)
Video Sources	DH	7	8-bit uns. int.		(2)
Overlay Service	DH	8	8-bit uns. int.	1 = Available 0 = Not available	(3)
Graphic Plane Intensity	DH	9	8-bit uns. int.		
Video Plane Intensity	DH	10	8-bit uns. int.		
Dissolve Level	DH	11	8-bit uns. int.		
Transparency Palette	DH	12	8-bit uns. int.	1 = Enabled 0 = Disabled	
First Color	DH	13-14	16-bit uns. int.		(4)
Last Color	DH	15-16	16-bit uns. int.		(4)
Palette Address	DH	17-20	32-bit address		(5)

Notes

1. Returns the number of logical colors that can be displayed simultaneously on the graphic plane.
2. Returns the number of video sources available for display on the video plane.
3. Returns whether or not overlay services are available in the current graphic mode.
4. Specifies the upper and lower bounds of a range of logical colors whose current transparency palette entries will be returned by this service.
5. The address of an array of transparency palette entries for the range of logical colors specified by the previous two fields. Each element of this array returns an 8-bit, unsigned integer that specifies the current transparency palette entry for a logical color.

Name **Set Colors Translucent (Extended)**

Service ID 13

Description Sets the transparency palette entries for logical colors on the graphic plane. This extended service allows values other than 0 (transparent) and 255 (opaque).

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	13	
Block Length	A	2-3	16-bit uns. int.	13	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 5 = Palette bound out of range	
First Color	A	5-6	16-bit uns. int.		(1)
Last Color	A	7-8	16-bit uns. int.		(1)
Palette Address	A	9-12	32-bit address		(2)

Notes

1. Specifies the upper and lower bounds of a range of logical colors whose transparency palette entries will be changed by this service.
2. The address of an array of new transparency palette entries for the range of logical colors specified by the previous two fields. Each element of this array must be an 8-bit unsigned integer that specifies the transparency palette entry for a logical color, where 255 means opaque and 0 means transparent and values between specify a mixture of graphic and video.

Name **Set Video Intensity (Extended)**

Service ID 14

Description Sets the video plane intensity over a specified amount of time. This extended service allows values other than 0 (off) and 255 (on). During asynchronous operation, all other services operate as specified.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	14	
Block Length	A	2-3	16-bit uns. int.	9	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding	
Video Intensity	A	5	8-bit uns. int.		
Duration	A	6-7	16-bit uns. int.		(1)
Sync/ Async	A	8	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(2)

Notes

1. Specifies the time, in seconds, during which the video plane will change from its current intensity to the new intensity. If the time specified is zero, the video plane is set to the new intensity immediately. If a non-zero time is specified, the video plane will change from its current intensity to the new intensity uniformly over the time specified.

2. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," is issued when the video plane is at the new intensity.

Name **Set Graphic Intensity (Extended)**

Service ID 15

Description Sets the graphic plane intensity over a specified amount of time. This extended service allows values other than 0 (off) and 255 (on). During asynchronous operation, all other services operate as specified.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	15	
Block Length	A	2-3	16-bit uns. int.	9	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding	
Graphic Intensity	A	5	8-bit uns. int.		
Duration	A	6-7	16-bit uns. int.		(1)
Sync/ Async	A	8	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(2)

Notes

1. Specifies the time, in seconds, during which the graphic plane will change from its current intensity to the new intensity. If the time specified is zero, the graphic plane is set to the new intensity immediately. If a non-zero time is specified, the graphic plane will change from its current intensity to the new intensity uniformly over the time specified.

2. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," is issued when the graphic plane is at the new intensity.

Name **Set Dissolve (Extended)**

Service ID 16

Description Sets the dissolve level over a specified amount of time. During asynchronous operation, all other services operate as specified.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	16	
Block Length	A	2-3	16-bit uns. int.	9	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding	
Dissolve Level	A	5	8-bit uns. int.		
Duration	A	6-7	16-bit uns. int.		(1)
Sync/ Async	A	8	8-bit uns. int.	0 = Asynchronous 1 = Synchronous	(2)

Notes

1. Specifies the time, in seconds, during which the dissolve level will change from its current level to the new level. If the time specified is zero, the new level is set immediately. If a non-zero time is specified, the level will change from its current level to the new level uniformly over the time specified.
2. Specifies whether this service will be executed synchronously or asynchronously. When executed asynchronously, alert 0, "Service Complete," is issued when the dissolve level is at its new value.

Name **Timed Service Completed**

Alert ID 0

Description Issued when an asynchronous, timed service has completed.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Alert ID	DH	0-1	16-bit uns. int.	0
Block Length	DH	2-3	16-bit uns. int.	6
ID of Completed Service	DH	4-5	8-bit uns. int.	14 = Set video intensity 15 = Set graphic intensity 16 = Set dissolve

Name **Graphic Plane Mode Changed**

Alert ID 1

Description Issued whenever the number of logical colors or the pixel resolution on the graphic plane changes because of a mode switch on the graphic device. This alert also informs the application whether or not the overlay device is capable of providing overlay services in the new graphic mode.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Alert ID	DH	0-1	16-bit uns. int.	1	
Block Length	DH	2-3	16-bit uns. int.	7	
Palette Size	DH	4-5	16-bit uns. int.		(1)
Overlay Service Availability	DH	6	8-bit uns. int.	1 = Available 0 = Not available	

Note

1. Returns the number of logical colors that can be displayed at one time on the graphic plane after the mode change.

APPENDIX D

AUDIO MANAGEMENT SERVICES

Name **Initialize Device Handler**

Service ID 0

Description Initializes the device handler and its physical device. Invoked before any other service. (Multiple invocations should not affect the device.) Described in Section 4 of the main body of this document.

Block Format

Field Name	Set By[a]	Byte #	Format[b]	Possible Values	Notes
Service ID	RC	0-1	16-bit uns. int.	0	
Block Length	RC	2-3	16-bit uns. int.	19	
Status	DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 7 = Already initialized	(1)
Init String Address	RC	5-8	32-bit address		(2)
R&C Alert Address	RC	9-12	32-bit address		(3)
Available Memory Bytes	RC	13-16	32-bit uns. int.		(4)
Next Segment	DH	17-18	16-bit address		(5)

[a] Throughout Appendix D, this column indicates which program(s) set the specified field: A = application; RC = R&C program; DH = device handler.
[b] Throughout Appendix D, the abbreviation "uns. int." = unsigned integer.

Notes

1. The device handler should take no action when this service is invoked multiple times.
2. The syntax and semantics of a device handler's initialization string are dictated by the device handler developer. The initialization string appears in the DSI configuration file (see Thomason and Van de Wetering (1990) in the main body of this document).
3. Address of R&C program entry point that routes alert packets to the application.
4. Unused memory that is contiguous to the device handler and available for its use.
5. Set by device handler. Address where the R&C program may load another device handler without interfering with subject handler. See Section 3 in the main body of this document.

Name **Terminate Device Handler**

Service ID 1

Description Prepare the physical peripheral to be powered off. Restore any system interrupt vectors altered during device handler initialization and operation. This service immediately overrides any other command.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	RC	0-1	16-bit uns. int.	1
Block Length	RC	2-3	16-bit uns. int.	5
Status	DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error

Name **Services Query**

Service ID 2

Description Lists all services and alerts provided by the target device. Returns two bit-vectors, one describing services and one describing alerts. The N'th bit in each vector reflects support for the N'th service or alert. Allows an application to determine which extended services are supported by a particular device.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	2	
Block Length	A	2-3	16-bit uns. int.	15	
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy	
Request Vector Byte Length	DH	5	8-bit uns. int.		
Request Vector Address	DH	6-9	32-bit address to bit-vector		(1)
Alert Vector Byte Length	DH	10	8-bit uns. int.		
Alert Vector Address	DH	11-14	32-bit address to bit-vector		(1)

Note

1. Bits set to 1 indicate a supported service. Bits set to 0 reflect an unsupported service.

Name **Set Alert Mask**

Service ID 3

Description Enables and disables alerts based on the contents of a bit-vector, called the alert mask. This mask has the same format as service 2's alert vector. Included for future services. No alerts are currently specified for this device.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	3
Block Length	A	2-3	16-bit uns. int.	10
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy
Alert Mask Byte Length	A	5	8-bit uns. int.	
Alert Mask Address	A	6-9	32-bit address to bit-vector	

Name **Disable All Alerts**

Service ID 4

Description Disables all alerts. Included for future services. No alerts are currently specified for this device.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	4
Block Length	A	2-3	16-bit uns. int.	5
Status	RC/ DH	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy

Name **Audio Management Attributes**

Service ID 5

Description Returns information about the configuration of the audio management device.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	5
Block Length	A	2-3	16-bit uns. int.	7
Status	DH/RC	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding
Sources Available	DH	5	8-bit uns. int.	
Outputs Available	DH	6	8-bit uns. int.	

Name **Audio Source On/Off**

Service ID 6

Description Turns the selected audio source on or off.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	6
Block Length	A	2-3	16-bit uns. int.	7
Status	DH/ RC	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 5 = Source out of range
Source	A	5	8-bit uns. int.	
Source On/Off	A	6	8-bit uns. int.	0 = Off 1 = On

Name **Audio Output On/Off**

Service ID 7

Description Turns the selected audio output on or off.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	7
Block Length	A	2-3	16-bit uns. int.	7
Status	DH/ RC	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 6 = Output out of range
Output	A	5	8-bit uns. int.	
Output On/Off	A	6	8-bit uns. int.	0 = Off 1 = On

Name **Set Source Level**

Service ID 8

Description Sets the selected audio source's level.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	8
Block Length	A	2-3	16-bit uns. int.	8
Status	DH/ RC	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 5 = Source out of range 6 = Output out of range
Source	A	5	8-bit uns. int.	
Output	A	6	8-bit uns. int.	
Level	A	7	8-bit uns. int.	

Name **Set Output Level**

Service ID 9

Description Sets the selected audio output's level.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	9
Block Length	A	2-3	16-bit uns. int.	7
Status	DH/ RC	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 6 = Output out of range
Output	A	5	8-bit uns. int.	
Level	A	6	8-bit uns. int.	

Name **Query Source State**

Service ID 10

Description Returns information about the current state of the specified audio source.

Block Format

Field Name	Set By	Byte #	Format	Possible Values
Service ID	A	0-1	16-bit uns. int.	10
Block Length	A	2-3	16-bit uns. int.	7
Status	DH/ RC	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 5 = Source out of range
Source	A	5	8-bit uns. int.	
Source On/Off	DH	6	8-bit uns. int.	0 = Off 1 = On

Name **Query Output State**

Service ID 11

Description Returns information about the current state of the specified audio output.

Block Format

Field Name	Set By	Byte #	Format	Possible Values	Notes
Service ID	A	0-1	16-bit uns. int.	11	
Block Length	A	2-3	16-bit uns. int.	13	
Status	DH/ RC	4	8-bit uns. int.	0 = Service successful 1 = Unknown error 2 = Device not available 3 = Device handler busy 4 = Hardware not responding 6 = Output out of range	
Output	A	5	8-bit uns. int.		
Level	DH	6	8-bit uns. int.		
Output On/Off	DH	7	8-bit uns. int.	0 = Off 1 = On	
Source Levels Array Length	DH	8	8-bit uns. int.		
Source Levels Array	DH	9-12	32-bit address		(1)

Note 1. Returns the address of an array containing the source level values for the mixture produced by the selected output.

Distribution List

Distribution:
Assistant Secretary of Defense (Force Management and Personnel)
Deputy Under Secretary of Defense for Research and Engineering (Research and Advanced Technology)
Director, Total Force Training and Education (OP-11)
Director, Aviation Training Systems Program Coordinator (PMA-205)
Commanding Officer, Naval Training Systems Center
Defense Technical Information Center (DTIC) (2)

Copy to:
Deputy Chief of Naval Operations (MP&T) (OP-01)
Assistant for Planning and Technical Development (OP-01B2)
Head, Training and Education Assessment (OP-11B1)
Director, Total Force Information System Management (OP-16)
Director, Submarine Manpower and Training Requirements (OP-29)
Director, Command Surface Warfare Manpower and Training Requirements (OP-39)
Director, Air ASW Training (OP594)
Assistant for Manpower, Personnel, and Training (OP-983D)
Commander, Naval Sea Systems Command (PMS 350)
Commander, Naval Sea Systems Command (PMS 396)
Commander, Naval Sea Systems Command (CEL-MP)
Director, Strategic Systems Project (SP-15)
Commanding Officer, New London Laboratory, Naval Underwater Systems Center (Code 33A)
Commanding Officer, New London Laboratory, Naval Underwater Systems Center (Code 3333)
Naval Training Systems Center, Technical Library (5)
Naval Training Systems Center (Code 10), (Code N-1), (Code 7)
Director, Office of Naval Research (OCNR-10)
Chief Scientist, Office of Naval Technology (OCNR-20T)
Chief of Naval Education and Training (Code 00)
Director, Training Technology (Code N-54)
Commanding Officer, Naval Education and Training Program Management Support Activity (Code 03) (2)
Commanding Officer, Naval Education and Training Program Management Support Activity (Code 04) (2)
Chief of Naval Technical Training (Code 00) (2)
Commander, Naval Military Personnel Command (NMPC-00/PERS-1)
Naval Military Personnel Command, Library (Code NMPC-013D)
Commanding Officer, Naval Health Sciences Education and Training Command, Bethesda, MD
Commander, Naval Reserve Force, New Orleans, LA
Commandant of the Marine Corps, Commanding General, Marine Corps Research and Development and Acquisition Command
Commander, U.S. ARI, Behavioral and Social Sciences, Alexandria, VA (PERTI-POT-I)
Technical Director, U.S. ARI, Behavioral and Social Sciences, Alexandria, VA (PERI-ZT)
Commander, Air Force Human Resources Laboratory, Brooks Air Force Base, TX

Scientific and Technical Information (STINFO) Office
TSRL/Technical Library (FL 2870)
Program Manager, Life Sciences Directorate, Bolling Air Force Base, DC (AFOSR/NL)
Commander, OPSTNGDIV Air Force Human Resources Laboratory, Williams Air Force Base, AZ (AFHRL/OT)
Commander, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base, OH, Logistics and Human Factors Division (AFHRL/LRS-TDC)
Director of Training, Office of Civilian Personnel Management
Superintendent, Naval Postgraduate School
Director of Research, U.S. Naval Academy
Center for Naval Analyses